Manual de Referencia del Identificador Gráfico de Bifurcaciones

Andrés Tovar Pérez
Profesor Asociado
Departamento de Ingeniería Mecánica y Mecatrónica
Universidad Nacional de Colombia

http://www.unal.edu.co/optimun/igb

Última revisión, Abril de 2007

Contenido

1. Introducción

El Identificador Gráfico de Bifurcaciones IGB es una herramienta creada para localizar valores críticos de parámetros independientes en sistemas dinámicos autónomos multiparamétricos continuos y discretos. Se ha diseñado para identificar bifurcaciones locales de codimensión 1, pero cuenta también con herramientas para la localización y visualización de bifurcaciones de codimensión mayor.

En el IGB se han implementado métodos directos (funciones de prueba) y métodos indirectos (continuación) teóricamente fundamentados en el capítulo 6. Estos se han probado con funciones tipo que describen las formas normales de las bifurcaciones de codimensión 1: *fold* y Hopf para sistemas continuos, y *fold*, *flip* y Neimark – Sacker para sistemas discretos.

El IGB ha sido diseñado como un *toolbox* para MATLAB. Sus funciones, además de operar por si solas, son utilizadas desde una interfaz gráfica descrita en detalle en el Manual del Usuario.

En este Manual de Referencia se presentan las funciones del IGB de las que se explican los siguientes aspectos:

Propósito	Breve descripción de la función
Sintaxis	Muestra el formato de la función
Descripción	Describe lo que hace la función, las reglas para su uso y sus limitaciones
Ejemplo	Se muestran uno o varios ejemplos para su aplicación
Algoritmo	Algoritmo o método numérico central de la función
Ver también	Refiere a otras funciones relacionadas
Referencias	Documentación para ampliar explicación

El Manual de Referencia se divide en las siguientes secciones:

Tablas de Referencia	Contiene la descripción resumida de cada una de las funciones del IGB
Funciones básicas	Estas funciones son la base para la localización de equilibrios y puntos fijos. Son las que se utilizan para corregir las predicciones en la localización de ramales en sistemas multiparamétricos y no paramétricos.
Métodos indirectos básicos	Son funciones que utilizan a su vez las funciones básicas para la localización de ramales. También incluyen las funciones para la simulación del comportamiento del sistema dinámico.
Métodos indirectos de continuación	En estas funciones se han implementado las técnicas de continuación basadas en el método de predictor – corrector con y sin parametrización, para sistemas dinámicos continuos y discretos.
Funciones de prueba	Los métodos directos para la identificación del tipo de bifurcación se basan en las funciones de prueba desarrolladas para el IGB.
Funciones del sistema dinámico	El sistema dinámico a analizar, debe ser descrito de una forma tal que pueda ser interpretado por IGB. En esta sección se presentan las plantillas y la sintaxis que se debe seguir en su descripción.

El Manual del Usuario, que se encuentra en al capítulo siguiente, recoge las instrucciones para la instalación del IGB y para la operación de la funciones desde su interfaz.

2. Tablas de referencia

Las funciones del IGB se encuentra agrupadas por categorías de acuerdo con las características de los algoritmos utilizados para su creación y uso. Estas categorías son: funciones básicas, funciones gráficas, métodos de continuación, funciones de prueba y funciones del sistema dinámico.

Funciones básicas	
NEWTON	Método de Newton para la localización de equilibrios
NEWTONMP	Método de Newton para la localización de equilibrios en sistemas multiparamétricos
SECANT	Método secante para la localización de equilibrios
SECANTMP	Método secante para la localización de equilibrios en sistemas multiparamétricos
SEIDEL	Método de Seidel para la localización de puntos fijos
SEIDELMP	Método de Seidel para la localización de puntos fijos en sistemas multiparamétricos
BOLZANO	Método de bisección de Bolzano para la localización de equilibrios

Funciones gráficas	
RFASE	Retrato de fase del sistema
XVST	Gráfico de tiempo (solo en IGB ver. 5)
FBRUTA	Identificación de bifurcaciones por fuerza bruta
PUNI	Continuación con corrector Newton – Raphson para sistemas continuos
PUNID	Continuación con corrector Newton – Raphson para sistemas discretos
PMUL	Corrector Newton – Raphson a partir de varios puntos en sistemas continuos
PMULD	Corrector Newton – Raphson a partir de varios puntos en sistemas discretos
PSECANT	Continuación con corrector Secante para sistemas continuos
PSECANTD	Continuación con corrector Secante para sistemas discretos
PSEIDEL	Continuación con corrector Seidel para sistemas discretos

Métodos de continuación	
CSC	Continuacion con predictor secante para sistemas continuos
CSCD	Continuacion con predictor secante para sistemas discretos
CSCP	Continuacion con predictor secante para sistemas continuos parametrizados
CSCPD	Continuacion con predictor secante para sistemas discretos parametrizados
CTC	Continuacion con predictor tangente para sistemas continuos
CTCD	Continuacion con predictor tangente para sistemas discretos
CTCP	Continuacion con predictor tangente para sistemas continuos parametrizados
CTCPD	Continuacion con predictor tangente para sistemas discretos parametrizados

Funciones de prueba	
TFUN	Función de prueba para bifurcaciones en sistemas continuos
TFUND	Función de prueba para bifurcaciones en sistemas discretos

Funciones del sistema dinámico	
`FUN`	Plantilla para la función del sistema continuo
`GRADFUN`	Plantilla para el Jacobiano de la función del sistema continuo
`FUNP`	Plantilla para la función parametrizada del sistema continuo
`GRADFUNP`	Plantilla para el Jacobiano de la función parametrizada del sistema continuo
`FUND`	Plantilla para la función del sistema discreto
`GRADFUND`	Plantilla para el Jacobiano de la función del sistema discreto

3. Funciones básicas

Las funciones básicas son la base para la localización de equilibrios y puntos fijos. Por lo general se utilizan para corregir las predicciones en la localización de ramales en sistemas multiparamétricos y no paramétricos. Dentro de las funciones básicas se encuentran métodos de Newton – Raphson, secante, Seidel y Bolzano.

Para la localización de equilibrios en sistemas dinámicos continuos, se han implementado los métodos de Newton – Raphson y Secante en las siguientes funciones:

- Newton
- NewtonMp
- Secant
- SecantMp

Para la localización de puntos fijos, se ha implementado el método de Seidel en la función:

- Seidel
- SeidelMp

Un método de utilidad para la detección bien de un punto fijo o de un equilibrio, es el de Bolzano implementado en la siguiente función:

- Bolzano

3.1 Newton

3.1.1 Propósito

Resolver un sistema de ecuaciones no lineales de la forma $f(x) = 0$, a partir de un vector inicial x_0.

3.1.2 Sintaxis

```
[X,F] = newton(FUN,GRADFUN,X0,DELTA,EPSILON)
```

donde

```
F(X) = 0.
```

3.1.3 Descripción

`[X,F]=NEWTON('FUN','GRADFUN',X0)` inicia en el vector `X0` e intenta resolver el vector de ecuaciones descrito en el archivo `FUN`. `FUN` es un archivo `.M` que calcula el valor de `X`: `F=FUN(X)`.

`GRADFUN` es el nombre del archivo `.M`, que contiene el Jacobiano de la función `FUN`, y que calcula las derivadas `dF/dX` en `X`: `GF=GRADFUN(X)`.

`[X,F]=NEWTON('FUN','GRADFUN',X0,DELTA,EPSILON)` permite especificar las tolerancias `DELTA` y `EPSILON` para el criterio de convergencia por tamaño de paso `DX` y por valor de la función `F(X)` respectivamente. Por defecto estos valores son 10^(-5).

Esta función está limitada a sistemas dinámicos continuos de la forma $f(x) = 0$. No admite sistemas con parámetros independientes. Para ser usada con sistemas dinámicos discretos estos deben plantearse como un sistema continuo de la forma $f(x) = x - g(x)$, donde $x_{k+1} = g(x_k)$.

3.1.4 Ejemplo

Para una función $f(x) = x^2 - 2x - 1$, descrita en el archivo `test1.m`, así

```
function fdx=test1(x)
    fdx=x^2-2*x-1;
```

y cuyo Jacobiano $fp(x) = 2x - 2$ se describe en el archivo `test1j.m`, así

```
function fpdx=test1j(x)
    fpdx=2*x-2;
```

es posible utilizar el método de Newton – Raphson para determinar la raíz de $f(x)$, a partir de un punto inicial $x_0 = 4$, por ejemplo. La sintaxis es la siguiente

```
[x,f]=newton('test1','test1j',4)
```

Se obtendrá como resultado

```
la solución fue encontrada dentro de las tolerancias
x=2.41421356237311
f=4.707345624410663e-014
```

3.1.5 Algoritmo

Utiliza la iteración

$$x_k = x_k - \frac{f(x_{k-1})}{f'(x_{k-1})}, \text{ para } k = 1, 2, ...$$

Donde $f'(x)$ es el Jacobiano de la función.

3.1.6 Ver también

`NEWTONMP, SECANT, SECANTMP`

3.1.7 Referencias

Capítulo 5.

3.2 NewtonMp

3.2.1 Propósito

Resolver un sistema de ecuaciones no lineales con parámetros independientes, de la forma $f(x, \alpha) = 0$, a partir de un vector inicial $y_0 = [x_0, \alpha]$

.

3.2.2 Sintaxis

```
function [Y,F,k,cond] = newtonmp(FUN,GRADFUN,Y0,PPAR,DELTA,EPSILON)
```

donde

```
F(Y) = 0.
```

3.2.3 Descripción

`[X,F,k,cond]=NEWTONMP('FUN','GRADFUN',X)` inicia en el vector `X`, e intenta resolver el vector de ecuaciones descrito en el archivo `FUN`. `GRADFUN` es el nombre del archivo, que contiene el Jacobiano de la función `FUN`, y que calcula las derivadas `dF/dX` en `X`: `GF=GRADFUN(X)`.

`F` permite conocer el valor de la función en el punto `X` encontrado. `k` permite conocer el número de iteraciones utilizado. `cond` permite conocer la condición del cálculo:

Si `cond=0`, el número máximo de iteraciones fue excedido.
Si `cond=1`, el determinante del Jacobiano es cero.
Si `cond=2`, la solución fue encontrada dentro de las tolerancias.

`NEWTONMP('FUN','GRADFUN',X,PPAR)` permite identificar las posiciones de los parámetros independientes `[alfa1,alfa2,...]`. `PPAR` es un vector columna que contiene las posiciones.

`NEWTONMP('FUN','GRADFUN',X,PPAR,DELTA,EPSILON)` permite especificar las tolerancias `DELTA` y `EPSILON` para el criterio de convergencia por tamaño de paso

`DX` y por valor de la función `F(X)` respectivamente. Por defecto estos valores son 10^(-5).

3.2.4 Ejemplo

Para la función $f(x) = \alpha + x^2$, descrita en el archivo `fold.m`, así

```
function yp=fold(y)
x=y(1); alfa=y(2);
yp=alfa + x^2;
```

y cuyo Jacobiano es $fp(x) = 2x$, descrito en el archivo `foldj.m`, así

```
function J=foldj(y)
x=y(1); alfa=y(2);
J=2*x;
```

Es posible utilizar el método de Newton – Raphson multiparamétrico para determinar un equilibrio del sistema a partir de un punto inicial y = (-0.5,-0.5), por ejemplo. La sintaxis es la siguiente

```
[x,f,k,cond]=newtonmp('fold','foldj',[-0.5,-0.5],2)
```

Se obtendrá como resultado

```
x=[-0.70710678118734;-0.50000000000000]
f=1.127653526111772e-012
k=4
cond=2
```

3.2.5 Algoritmo

Utiliza la iteración

$$x_k = x_k - \frac{f(x_{k-1})}{f'(x_{k-1})}, \text{ para } k = 1, 2, ...$$

Donde $f'(x)$ es el Jacobiano de la función, teniendo la precaución de no variar el valor de los parámetros independientes.

3.2.6 Ver también

NEWTON, SECANT, SECANTMP

3.2.7 Referencias

Capítulo 5.

3.3 Secant

3.3.1 Propósito

Resolver un sistema de ecuaciones no lineales de la forma $f(x) = 0$, a partir de dos puntos iniciales x_0 y $x_1 = x_0 + s$.

3.3.2 Sintaxis

```
function [X,F] = secant(FUN,X,s,DELTA,EPSILON)
```

3.3.3 Descripción

`[X,F]=SECANT('FUN',X,s)` iniciando con los puntos `X` y `X+s` e intenta resolver el vector de ecuaciones descrito en el archivo `FUN`. `F=FUN(X)`.

`[X,F]=SECANT('FUN',X,s,DELTA,EPSILON)` permite especificar las tolerancias `DELTA` y `EPSILON` para el criterio de convergencia por tamaño de paso `DX` y por valor de la función `F(X)` respectivamente. Por defecto estos valores son 10^(-5).

A pesar de que no necesita la definición del Jacobiano del la función, tal como en los métodos de Newton – Rapshon, el método Secante no es apropiado para sistemas de más de dos dimensiones.

3.3.4 Ejemplo

Para una función $f(x) = x^2 - 2x - 1$, descrita en el archivo `test1.m`, así

```
function fdx=test1(x)
    fdx=x^2-2*x-1;
```

es posible utilizar el método Secante para determinar la raíz de $f(x)$, a partir de los puntos iniciales $x_0 = 4$ y $x_1 = 4.1$, por ejemplo. La sintaxis es la siguiente

```
[x,f]=secant('test1',4,0.1)
```

Se obtendrá como resultado

```
la solución fue encontrada dentro de las tolerancias
x=2.41421356452856
f=6.096587235049355e-009
```

3.3.5 Algoritmo

Utiliza la iteración

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

3.3.6 Ver también

SECANTMP, NEWTON, NEWTONMP

3.3.7 Referencias

Capítulo 5 .

3.4 SecantMp

3.4.1 Propósito

Resolver un sistema de ecuaciones no lineales con parámetros independientes, de la forma $f(x, \alpha) = 0$, a partir de dos puntos iniciales $y_0 = [x_0, \alpha]$ y $y_1 = y_0 + s$.

3.4.2 Sintaxis

```
function [Y,F,k,cond] = secantmp(FUN,Y0,s,PPAR,DELTA,EPSILON)
```

donde,

```
F(Y) = 0.
```

3.4.3 Descripción

`[X,F,k,cond]=SECANTMP('FUN',Y0,s)` iniciando con los puntos `Y0` y `Y0+s`. Intenta resolver el vector de ecuaciones descrito en el archivo `FUN` por el método secante.

`F` permite conocer el valor de la función en el punto `X` encontrado. `k` permite conocer el número de iteraciones utilizado. `cond` permite conocer la condición del cálculo:

Si `cond=0`, el número máximo de iteraciones fue excedido
Si `cond=1`, se encontró una división por cero
Si `cond=2`, la solución fue encontrada dentro de las tolerancias

`SECANTMP('FUN',Y0,s,PPAR)` permite identificar la posición de los parámetros independientes `[alfa1,alfa2,...,alfam]` cuando el sistema los tiene. Estas posiciones se indican en el vector `PPAR`.

`SECANTMP('FUN',Y0,s,PPAR,DELTA,EPSILON)` permite especificar las tolerancias `DELTA` y `EPSILON` para el criterio de convergencia por tamaño de paso `DX` y por valor de la función `F(X)` respectivamente. Por defecto estos valores son 10^(-5).

A pesar de que no necesita la definición del Jacobiano del la función, tal como en los métodos de Newton – Rapshon, el método Secante no es apropiado para sistemas de más de dos dimensiones.

3.4.4 Ejemplo

Para la función $f(x) = \alpha + x^2$, descrita en el archivo `fold.m`, así

```
function yp=fold(y)
x=y(1); alfa=y(2);
yp=alfa + x^2;
```

Es posible utilizar el método Secante multiparamétrico para determinar un equilibrio del sistema a partir de los puntos iniciales y_0 = (-0.50,-0.50) y y_1 = (-0.45, -0.45), por ejemplo. La sintaxis es la siguiente

```
[x,f,k,cond]=secantmp('fold',[-0.5,-0.5],0.05,2)
```

Se obtendrá como resultado

```
x=[-0.70710678072253,-0.50000000000000]
f=-6.562161369849661e-010
k=9
cond=2
```

3.4.5 Algoritmo

Utiliza la iteración

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

En cada iteración deja el valor del parámetro independiente como el del punto inicial.

3.4.6 Ver también

`SECANT, NEWTON, NEWTONMP`

3.4.7 Referencias

Capítulo 5 .

3.5 Seidel

3.5.1 Propósito

`SEIDEL` Resuelve sistemas de ecuaciones no lineales utilizando el método de Seidel. Localiza puntos fijos en sistemas dinámicos discretos de la forma

$$x \mapsto g(x)\ ,\ \ x \in \mathbf{R}^n$$

3.5.2 Sintaxis

```
function [X,F]=seidel(FUN,X0,TOL)
```

donde

```
FUN(X)=X
```

3.5.3 Descripción

`[X,F]=SEIDEL('FUN',X0)` inicia en el vector `X0` e intenta resolver el vector de ecuaciones descrito en el archivo `FUN`. `FUN` es un archivo que contiene el valor de `F=FUN(X).`

`[X,F]=SEIDEL('FUN',X0,TOL)` permite especificar la tolerancia `TOL` para el criterio de convergencia. Por defecto `TOL` = 10^(-5).

Aunque es un método desarrollado para resolver funciones discretas de cualquier orden, no puede resolver funciones con parámetros independientes. Para esto se ha creado el método `SEIDELMP`. Con el método de Seidel solo se pueden identificar puntos fijos estables.

3.5.4 Ejemplo

Para la función

$$x = \frac{x^2 - y + 0.5}{2}$$

$$y = \frac{-x^2 - 4y^2 + 8y + 4}{8}$$

descrita en el archivo `test2.m` como:

```
function fdx=test2(x)
p=x(1); q=x(2);
fdp=(p^2-q+0.5)/2;
fdq=(-p^2-4*q^2+8*q+4)/8;
fdx=[fdp,fdq];
```

Es posible determinar un punto fijo del sistema con el método de Seidel a partir del punto inicial [0,1], de la siguiente forma

```
[x,fx]=seidel('test2',[0;1])
La solución fue encontrada despues de 8 iteraciones.
x=[-0.22221280911430;0.99380851324262]
fx=[-0.22221475322798;0.99380851324262]
```

3.5.5 Algoritmo

El algoritmo utilizado en cada iteración es:

$fdx = f(x)$

for $i = 1$: length (fdx)

$x(i) = fdx(i)$

$fdx = f(x)$

end

3.5.6 Ver también

SEIDELMP, BOLZANO

3.5.7 Referencias

Capítulo 5.

3.6 SeidelMp

3.6.1 Propósito

SEIDELMP localiza puntos fijos en sistemas dinámicos discretos con múltiples parámetros independientes, de la forma:

$$(x,\alpha) \mapsto g(x)\ ,\quad x\in\mathbf{R}^n,\quad \alpha\in\mathbf{R}^m$$

3.6.2 Sintaxis

```
function [Y,F,k,cond]=seidelmp(FUN,Y0,PPAR,TOL)
```

donde

```
FUN(Y)=Y
```

3.6.3 Descripción

[Y,F,k,cond]=SEIDELMP('FUN',Y0) inicia en el vector Y0 e intenta resolver el vector de ecuaciones descrito en el archivo FUN, archivo que calcula el valor de Y: F=FUN(X).Esto es equivalente a utilizar la función SEIDEL('FUN',X0).

F permite conocer el valor de la función en el punto Y encontrado. k permite conocer el número de iteraciones utilizado. cond permite conocer la condición del cálculo:

Si cond=0, el número máximo de iteraciones fue excedido
Si cond=1, la solución fue encontrada dentro de las tolerancias

SEIDELMP('FUN',X0,PPAR) permite especificar la posición de los parámetros independientes [alfa1,alfa2,...] en el vector de entrada Y0. PPAR es un vector que contiene las posiciones.

SEIDELMP('FUN',X0,PPAR,TOL) permite especificar la tolerancia TOL para el criterio de convergencia. Por defecto TOL=10^(-5).

Con el método de Seidel solo se pueden determinar los puntos fijos estables.

3.6.4 Ejemplo

Para el sistema dinámico discreto que describe la forma normal de la bifurcación de Neimark – Sacker:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} \cos\theta(\alpha) & -\operatorname{sen}\theta(\alpha) \\ \operatorname{sen}\theta(\alpha) & \cos\theta(\alpha) \end{pmatrix} \left[(1+\alpha)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left(x_1^2 + x_2^2\right)\begin{pmatrix} a(\alpha) & -b(\alpha) \\ b(\alpha) & a(\alpha) \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right]$$

donde, $a(\alpha)=1$, $b(\alpha)=0$, y que se describe en el archivo `sackerd.m` así:

```
function yp=sackerd(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=1; b=0;
A=[cos(teta),-sin(teta);sin(teta),cos(teta)];
B=(1+alfa)*[x1;x2];
C=(x1^2+x2^2)*[a -b;b a]*[x1;x2];
yp=A*(B+C);
```

Se puede utilizar el método de Seidel multiparámetrico de la siguiente forma:

```
[y,f,k,cond]=seidelmp('sackerd',[0.5,0.5,-1],3)
y=[0.0000;0.0000;-1.0000]
f=1.0e-108*[0.0000;0.7515]
k=4
cond=1
```

3.6.5 Algoritmo

El algoritmo utilizado en cada iteración es:

$fdx = f(x)$
for $i = 1$: length (fdx)
    $x(i) = fdx(i)$
    $fdx = f(x)$
end

Teniendo en cuenta que los parámetros independientes deben permanecer constantes.

3.6.6 Ver también

SEIDEL, BOLZANO

3.6.7 Referencias

Capítulo 5.

3.7 Bolzano

3.7.1 Propósito

Esta función utiliza el método de la Bisección de Bolzano para encontrar una raíz de la ecuación `0=FUN(X)` en el intervalo `[A,B]`.

3.7.2 Sintaxis

```
function [C,cond]=bolzano(FUN,A,B,DELTA,EPSILON)
```

3.7.3 Descripción

`[C,cond]=BOLZANO('FUN',A,B)` recibe los puntos `A` y `B`, y la función `FUN` tales que `FUN(A)` y `FUN(B)` son de signo opuesto, y retorna la raíz `C` y la condición `cond`.

Si `cond=0`, el número máximo de iteraciones fue excedido
Si `cond=1`, los signos de `FUN(A)` y `FUN(B)` son iguales
Si `cond=2`, la solución fue encontrada sin problema

`[C,cond]=BOLZANO('FUN',A,B,DELTA,EPSILON)` permite especificar la tolerancia con la que se desea determinar la raíz. `DELTA` es la tolerancia por tamaño de paso y `EPSILON` por valor de la función. Por defecto estos valores son `10^(-5)`.

El método de la Bisección de Bolzano es apropiado solo para sistemas unidimensionales.

3.7.4 Ejemplo

Considerando la función $f(x) = \cos(x) + 1 - x$ definida en el archivo `test3.m` así,

```
function fdx=test3(x)
   fdx=cos(x)+1-x;
```

Se puede determinar una raíz en el intervalo [0.8, 1.6], por el método de la Bisección de Bolzano del siguiente modo,

```
[c,cond]=bolzano('test3',0.8,1.6)
c=1.4000
cond=2
```

3.7.5 Algoritmo

A partir de los puntos `A`, `B` que definen el intervalo, se determina la raíz `C` de $f(x)$ con unas tolerancias `DELTA` y `EPSILON`, mediante el siguiente algoritmo,

```
C=(A+B)/2;
if f(C) ≤ EPSILON;
        cond=2;
        se ha encontrado la raíz
elseif (sign(f(B))) = (sign(f(C)));
        B=C; f(B)=f(C);
else
        A=C; f(A)=f(C);
end
```

3.7.6 Ver también

`SEIDEL`

3.7.7 Referencias

MATHEWS, John. Numerical Methods for Mathematics, Science, and Engineering. Prentice Hall.

4. Funciones gráficas

Estas funciones generan gráficas básicas para el análisis del sistema. Estas gráficas son de dos tipos: *simulación* que muestra el comportamiento del sistema o trayectoria descrita por uno o varios puntos, y *diagramas de bifurcación* que muestran los equilibrio o puntos fijos.

Las primeras funciones gráficas implementadas para el análisis de sistemas dinámicos son las de simulación a partir de un punto. Estas gráficas puede presentarse en función del tiempo, o solo en función de las variables y parámetros del sistema. Las funciones que las incorporan son:

- RFase
- XvsT

El método de la fuerza bruta, que es también un método basado en simulación, ha sido implementado para la localización de bifurcaciones. La función que lo incorpora es:

- FBruta

Los métodos que utilizan el método de Newton-Raphson para la localización de los equilibrios a partir de múltiples puntos sobre el espacio son:

Para sistemas continuos

- PMul

Para sistemas discretos

- PMulD

Los métodos de continuación con corrector basado en el método de Newton-Raphson son:

Para sistemas continuos

- Puni

Para sistemas discretos

- PUniD

Los métodos de continuación con corrector basado en el método Secante son:

Para sistemas continuos

- PSecant

Para sistemas discretos

- PSecantD

El método de continuación con corrector basado en el método Seidel para sistemas discretos es:

- PSeidel

4.1 RFase

4.1.1 Propósito

RFASE genera el retrato de fase del sistema a partir de puntos indicados por el usuario.

4.1.2 Sintaxis

```
flag_stop=rfase(PLIM,FUN,y0,PPLT,PPAR,s,TOL,PREC,flag_sis)
```

4.1.3 Descripción

RFASE(PLIM,'FUN',Y) genera el reatrato de fase de la función FUN sistema a partir del punto inicial Y=[y1,y2,...].

PLIM especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup].

'FUN' es el nombre del archivo con extensión .M donde se encuentra definida la función del sistema continuo. Ver el archivo 'FUN.M'.

RFASE=(PLIM,'FUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

RFASE(PLIM,'FUN',Y,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...]. PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

RFASE(PLIM,'FUN',Y,PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es un vector de la misma dimesión de Y, que indica el tamaño del

primer paso. Su signo indica la dirección de este. Si S no es dado, se asume como de una centésima del eje horizontal.

RFASE(PLIM,'FUN',Y,PPLT,PPAR,S,TOL) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0). Por defecto este valor es 10^(-5).

RFASE(PLIM,'FUN',Y,PPLT,PPAR,S,TOL,PREC) permite especificar la precisión para la presentación de los puntos fijos o equilibrios localizados. Por defecnto PREC=4.

RFASE(PLIM,'FUN',Y,PPLT,PPAR,S,TOL,PREC,flag_sis) cuando el sistema dinámico es discreto se debe especificar un valor para flag_sis=2. Por defecto el sistema se considera continuo, osea flag_sis=1.

4.1.4 Ejemplo

El atractor de Lorenz definido por la función

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -\frac{8}{3} & 0 & y \\ 0 & -10 & 10 \\ -y & 28 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

se describe en el archivo LORENZ.M del siguiente modo:

```
function yp=lorenz(y)
A=[-8/3 0 0;0 -10 10;0 28 -1];
A(1,3)=y(2);
A(3,1)=-y(2);
yp=A*y;
```

Este atractor de Lorenz puede ser simulado con la función RFASE de la siguiente forma,

```
rfase([0 50;-25 25;-25 25],'Lorenz',[35 -10 -7],[1 2 3],0,0.01)
```

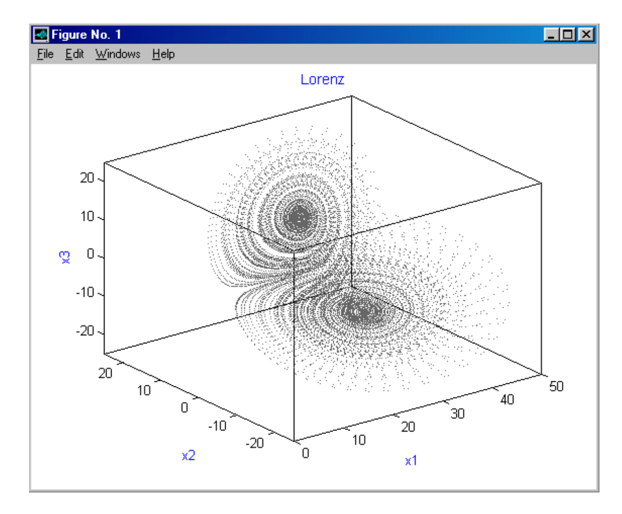


Figura 1 – Función RFASE en el IGB

Los datos graficados se consignan en el archivo `REGISTRO.TXT`.

4.1.5 Algoritmo

El algoritmo usado depende del tipo de sistema, para un sistema discreto simplemente calcula los puntos evaluando la función, así

$$x_{k+1} = g(x_k)$$

Para un sistema continuo (no autónomo), a partir de un punto fijo aproxima el siguiente así:

$$x_{k+1} = x_k + \Delta t \cdot f(x_k)$$

donde Δt es el tamaño de paso del tiempo seleccionado por el usuario.

4.1.6 Ver también

`XVST, FBRUTA`

4.1.7 Referencias

Capítulos 1 y 2.

4.2 XvsT

4.2.1 Propósito

XVST genera la gráfica de comportamiento del sistema en función del tiempo.

4.2.2 Sintaxis

```
flag_stop=xvst(PLIM,FUN,y0,PPLT,PPAR,s,TOL,PREC,flag_sis)
```

4.2.3 Descripción

XVST(PLIM,'FUN',Y) genera el reatrato de fase de la función FUN sistema a partir del punto inicial Y=[y1,y2,...].

PLIM especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup].

'FUN' es el nombre del archivo con extensión .M donde se encuentra definida la función del sistema continuo. Ver el archivo 'FUN.M'.

XVST=(PLIM,'FUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

XVST(PLIM,'FUN',Y,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...]. PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

XVST(PLIM,'FUN',Y,PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es un vector de la misma dimesión de Y, que indica el tamaño del primer paso. Su signo indica la dirección de este. Si S no es dado, se asume como

de una centésima del eje horizontal. La selección del tamaño de paso del tiempo resulta ser un factor crítico en la simulación del comportamiento en sistemas caóticos.

`XVST(PLIM,'FUN',Y,PPLT,PPAR,S,PREC)` permite especificar la precisión para la presentación de los puntos fijos o equilibrios localizados. Por defecnto `PREC=4`.

`XVST(PLIM,'FUN',Y,PPLT,PPAR,S,PREC,flag_sis)` cuando el sistema dinámico es discreto se debe especificar un valor para `flag_sis=2`. Por defecto el sistema se considera continuo, osea `flag_sis=1`.

4.2.4 Ejemplo

El atractor de Lorenz definido por la función

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -\frac{8}{3} & 0 & y \\ 0 & -10 & 10 \\ -y & 28 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

se describe en el archivo `LORENZ.M` del siguiente modo:

```
function yp=lorenz(y)
A=[-8/3 0 0;0 -10 10;0 28 -1];
A(1,3)=y(2);
A(3,1)=-y(2);
yp=A*y;
```

El comportamiento del atractor de Lorenz en el tiempo puede ser simulado con la función `XVST` de la siguiente forma,

```
xvst([0 80;-25 25],'Lorenz',[35 -10 -7],2,0,0.001)
```

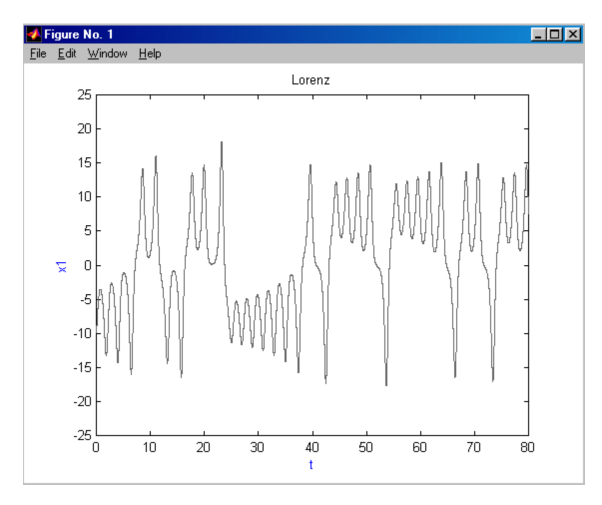


Figura 2 – Función XVST en el IGB

Presentando así el comportamiento de la primera variable de estado contra el tiempo. Los datos graficados se consignan en el archivo `REGISTRO.TXT`.

4.2.5 Algoritmo

El algoritmo usado depende del tipo de sistema, para un sistema discreto simplemente calcula los puntos evaluando la función, así

$$x_{k+1} = g(x_k)$$

Para un sistema continuo (no autónomo), a partir de un punto fijo aproxima el siguiente así:

$$x_{k+1} = x_k + \Delta t \cdot f(x_k)$$

donde Δt es el tamaño de paso del tiempo seleccionado por el usuario.

4.2.6 Ver también

`RFASE, FBRUTA`

4.2.7 Referencias

Capítulos 1 y 2.

4.3 FBruta

4.3.1 Propósito

FBRUTA genera el retrato de fase del sistema para la identificación de los equilibrios y puntos fijos estables mediante simulación.

4.3.2 Sintaxis

```
flag_stop=fbruta(PLIM,FUN,y0,PPLT,PPAR,s,...
         ITTOT,ITINI,PREC,flag_sis)
```

4.3.3 Descripción

FBRUTA(PLIM,'FUN',Y) realiza el proceso de continuación tangente con parametrización por longitud de arco de la función descrita en el archivo FUN a partir de un punto inicial Y=[y1,y2,...] calculando su derivada descrita en la función GRADFUN.

PLIM especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup].

FBRUTA=(PLIM,'FUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

FBRUTA(PLIM,'FUN',Y,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...] PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

FBRUTA(PLIM,'FUN',Y,PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es un vector de la misma dimensión de Y, que indica el tamaño

del primer paso. Su signo indica la dirección de este. Si `S` no es dado, se asume como de una centésima del eje horizontal.

`FBRUTA(PLIM,'FUN',Y,PPLT,PPAR,S,ITTOT,ITINI)` Permite especificar la tolerancia para localizar las soluciones de `0=FUN(y0)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

`FBRUTA(PLIM,'FUN',Y,PPLT,PPAR,S,ITTOT,ITINI,PREC)` permite especificar la precisión para la presentación de los puntos fijos o equilibrios localizados. Por defecto `PREC=4`.

`FBRUTA(PLIM,'FUN',Y,PPLT,PPAR,S,ITTOT,ITINI,PREC,flag_sis)` cuando el sistema dinámico es discreto se debe especificar un valor para `flag_sis=2`. Por defecto el sistema se considera continuo, o sea `flag_sis=1`.

4.3.4 Ejemplo

El sistema dinámico discreto definido como $x_{k+1} = \alpha(1 - x_k)x_k$, donde α es el parámetro independiente, se define de la siguiente forma en el archivo `logistic.m`:

```
function YP=logistic(Y)
    x=Y(1); R=Y(2);
    YP=R*(1-x)*x;
```

Puede ser estudiado con el método de la fuerza bruta para identificar visualmente las bifurcaciones que se presentan en el rango `[2.5 4]` para el parámetro independiente, a partir de puntos iniciales distintos a $x = 0$ (puntos fijos inestables del sistema). Para esto se ha seleccionado un tamaño de paso $s = 0.005$, y un número total de iteraciones de 500 de las cuales las primeras 250 no son graficadas.

```
fbruta([2.5 4;0 1],'logistic',[0.3 2.5],[2 1],2,0.005,500,250,4,2)
```

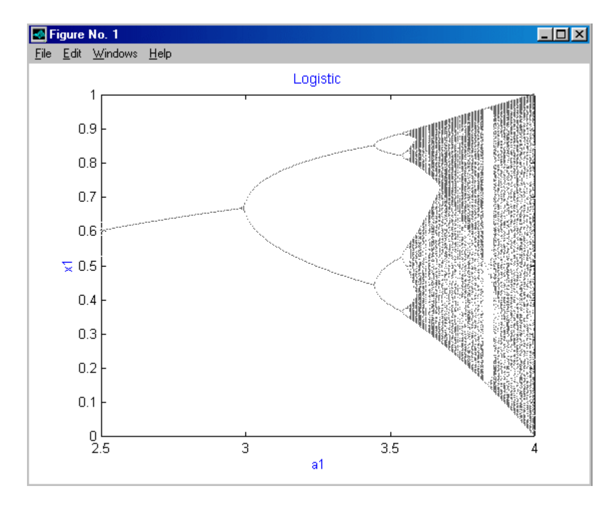


Figura 3 – Función FBRUTA en el IGB

Los datos graficados también se consignan en el archivo `REGISTRO.TXT`.

4.3.5 Algoritmo

```
yp=feval(FUN,y)
if flag_sis==1 %sistema continuo
      y=y+s*yp
elseif flag_sis==2 %sistema discreto
      y=yp
end
```

4.3.6 Ver también

`RFASE, PUNI, PUNID, PMUL, PMULD`

4.3.7 Referencias

Capítulos 5, 6 y 7.

4.4 PUni

4.4.1 Propósito

PUNI Identificación de Bifurcaciones mediante la localización de equilibrios a lo largo de un ramal por continuación con corrector basado en el método de Newton-Raphson a partir de un punto.

4.4.2 Sintaxis

```
flag_stop=puni(PLIM,FUN,GRADFUN,TFUN,y0,PPLT,PPAR,s,...
                   DELTA,EPSILON,PREC)
```

4.4.3 Descripción

PUNI(PLIM,'FUN','GRADFUN',’TFUN’,Y0) realiza el proceso de localización de equilibrios y puntos fijos mediante el método de Newonton – Raphson a partir del punto inicial Y0, en sistemas dinámicos multiparamétricos.

PLIM especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup].

FUN es el nombre del archivo con extensión M donde se encuentra definida la función. Ver el archivo FUN.M

GRADFUN es el nombre del archivo con extensión M donde se define el Jacobiano de la función. Ver el archivo GRADFUN.M

TFUN es el nombre del archivo con extensión M donde se encuentra definida la función de prueba. Ver el archivos TFUN.M y TFUND.M Si TFUN no está definido no podrá localizar bifurcaciones por el método directo de la función de prueba.

PUNI(PLIM,'FUN','GRADFUN',’TFUN’,Y0,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. Si PPLT no es dado, se plotearán las primeras variables.

PUNI(PLIM,'FUN','GRADFUN',’TFUN’,Y0,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...] PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

PUNI(PLIM,'FUN','GRADFUN',’TFUN’,Y0,PPLT,PPAR,S) permite especificas las distancias para calcular para determinar los puntos a partir de los cuales se determinarán los equilibrios (puntos fijos). S es un vector de dos o tres dimensiones cuyos componentes indican las distancias sobre cada uno de los ejes. Cuando S no está dado, se toma por defecto una fracción de 1/50 de los límites de la gráfica.

PUNI(PLIM,'FUN','GRADFUN',’TFUN’,Y0,PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

PUNI(PLIM,'FUN','GRADFUN',’TFUN’,Y0,PPLT,PPAR,S,DELTA,EPSILON,PREC) permite especificar la presentación para la representación de los equilibrios encontrados. Por defecto PREC=4.

4.4.4 Ejemplo

Si se define la forma normal de la bifurcación de Hopf y su Jacobiano en los archivos HOPF.M y HOPFJ.M del siguiente modo:

```
function yp=hopf(y)
x1=y(1); x2=y(2); alfa=y(3);
yp=[alfa*x1-x2-x1*(x1^2+x2^2);
x1+alfa*x2-x2*(x1^1+x2^2)];

function [J,ValPr,Z]=hopfj(y)
x1=y(1); x2=y(2); alfa=y(3);
J=[alfa-3*x1^2-x2^2, -1-2*x1*x2;
```

```
1-2*x1*x2, alfa-x1^2-3*x2^2];
        ValPr=eig(J);
         Z=[x1,x2]';
```

Se puede identificar una bifurcación de Hopf usando la función `PUNI` así:

```
puni([-1 1;-1 1;-1 1],'hopf','hopfj','tfun',[0,0,-1],...
[3,1,2],3,0.01)
```

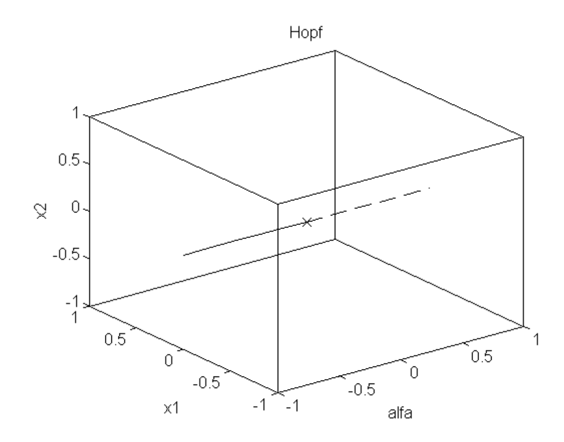


Figura 4 – Función PUNI en el IGB

El archivo `TFUN.M` contiene la definición de la función de prueba. Los equilibrios estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el equilibrio es indeterminado se grafica en gris (cruz).

4.4.5 Algoritmo

Utiliza el algoritmo del método de Newton – Raphson para la localización de los equilibrios. Comprueba la estabilidad evaluando el valor propio del equilibrio localizado.

4.4.6 Ver también

`PUNID, FUN, GRADFUN, TFUN, NEWTONMP`

4.4.7 Referencias

Capítulos 5, 6 y 7.

4.5 PUniD

4.5.1 Propósito

PUNID Identificación de Bifurcaciones mediante la localización de puntos fijos a lo largo de un ramal por continuación con corrector basado en el método de Newton-Raphson a partir de un punto.

4.5.2 Sintaxis

```
function flag_stop=punid(PLIM,FUN,GRADFUN,GRADFUND,TFUN,y0,...
                 PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

4.5.3 Descripción

PUNID Identificación de Bifurcaciones con el Método de Newton – Raphson a partir de un punto, para sistemas dinámicos discretos.

PUNID(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y0) realiza el proceso de localización de equilibrios y puntos fijos mediante el método de Newton - Raphson a partir del punto inicial Y0, en sistemas dinámicos multiparamétricos.

PLIM especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup].

FUN es el nombre del archivo con extensión M donde se encuentra definida la función del sistema continuo. Ver el archivo FUN.M

GRADFUN es el nombre del archivo con extensión M donde se define el Jacobiano de la función. Ver el archivo GRADFUN.M

GRADFUND es el nombre del archivo con extensión M donde se define el Jacobiano de la función discreta. Ver el archivo GRADFUND.M

`TFUN` es el nombre del archivo con extensión `M` donde se encuentra definida la función de prueba. Ver el archivos `TFUN.M` y `TFUND.M`. Si `TFUN` no está definido no podrá localizar bifurcaciones por el método directo de la función de prueba.

`PUNID(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y0,PPLT)`
permite identificar las posiciones de los parámetros que se desea plotear. Si `PPLT` no es dado, se plotearán las primeras variables.

`PUNID(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y0,PPLT,PPAR)`
permite identificar las posiciones de los parámetros independientes `[a1,a2,...]` `PPAR` es un vector columna que contiene sus posiciones. Si `PPAR` no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

`PUNID(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y0,PPLT,PPAR,S)`
permite especificas las distancias para calcular para determinar los puntos a partir de los cuales se determinarán los equilibrios (puntos fijos). `S` es un vector de dos o tres dimensiones cuyos componentes indican las distancias sobre cada uno de los ejes. Cuando `S` no está dado, se toma por defecto una fracción de 1/50 de los límites de la gráfica.

`PUNID(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y0,PPLT,PPAR,S,DELTA, EPSILON)` permite especificar la tolerancia para localizar las soluciones de `0=FUN(y0)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son `10^(-5)`.

`PUNID(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y0,PPLT,PPAR,S,DELTA, EPSILON,PREC)` permite especificar la precisión para la presentación de los equilibrios encontrados. Por defecto `PREC=4`.

4.5.4 Ejemplo

Si se define el Jacobiano de la forma normal de la bifurcación de Neimark – Sacker en el archivo `SACKERDJ.M`, y la expresión de esta función como sistema continuo y su Jacobiano en los archivos `SACKER.M` y `SACKERJ.M` del siguiente modo:

```
function yp=sacker(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=-alfa; b=0;
A=[cos(teta) -sin(teta);sin(teta) cos(teta)];
B=(1+alfa)*[x1;x2];
C=(x1^2+x2^2)*[a -b;b a]*[x1;x2];
yp=A*(B+C)-[x1;x2];
```

```
function [J,ValPr,Z]=sackerj(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=-alfa; b=0;
dAdx1=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(a*x1-b*x2);
dAdx2=   0    +   (x1^2+x2^2)*(-b)+ (2*x2)*(a*x1-b*x2);
dBdx1=   0    +   (x1^2+x2^2)*(b)+  (2*x1)*(b*x1+a*x2);
dBdx2=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(b*x1+a*x2);
I=[1 0;0 1];
J=[cos(teta)*dAdx1-sin(teta)*dAdx1,cos(teta)*dAdx2-sin(teta)*dAdx2;
sin(teta)*dBdx1+cos(teta)*dBdx1,cos(teta)*dBdx2-sin(teta)*dBdx2]-I;
dAdalfa=x1;dBdalfa=x2;
Z=[cos(teta)*dAdalfa-
sin(teta)*dAdalfa,sin(teta)*dBdalfa+cos(teta)*dBdalfa];
ValPr=eig(J);
```

```
function [J,ValPr]=SackerJD(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=-alfa; b=0;
dAdx1=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(a*x1-b*x2);
dAdx2=   0    +   (x1^2+x2^2)*(-b)+ (2*x2)*(a*x1-b*x2);
dBdx1=   0    +   (x1^2+x2^2)*(b)+  (2*x1)*(b*x1+a*x2);
dBdx2=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(b*x1+a*x2);
J=[cos(teta)*dAdx1-sin(teta)*dAdx1, cos(teta)*dAdx2-sin(teta)*dAdx2;
sin(teta)*dBdx1+cos(teta)*dBdx1,cos(teta)*dBdx2-sin(teta)*dBdx2];
ValPr=eig(J);
```

Se puede localizar la bifurcación de Neimark – Sacker utilizando la función `PUNID` del siguiente modo:

```
punid([-1 1;-1 1;-1 1],'sacker','sackerj','sackerdj','tfund',...
[0,0,-1],[3,1,2],3,0.01)
```

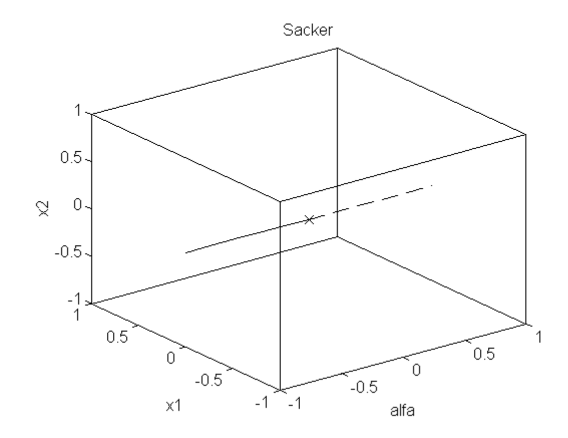


Figura 5 – Función PUNID en el IGB

El archivo `TFUND.M` contiene la definición de la función de prueba. Los puntos fijos estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el punto fijo es indeterminado se grafica en gris (cruz).

4.5.5 Algoritmo

Utiliza el algoritmo del método de Newton – Raphson para la localización de los equilibrios. Comprueba la estabilidad evaluando el valor propio del punto fijo localizado.

4.5.6 Ver también

`PUNI, FUN, GRADFUN, GRADFUND, TFUND, NEWTONMP`

4.5.7 Referencias

Capítulos 5, 6 y 7.

4.6 PMul

4.6.1 Propósito

PMUL Identificación de bifurcaciones mediante la identificación de equilibrios por aproximación a partir de múltiples puntos y solución con el Método de Newton-Raphson.

4.6.2 Sintaxis

```
function flag_stop=pmul(PLIM,FUN,GRADFUN,y0,PPLT,PPAR,s,...
                        DELTA,EPSILON,PREC)
```

4.6.3 Descripción

PMUL(PLIMS,'FUN','GRADFUN',Y0) realiza el proceso de localización a partir de puntos definidos dentro de los límites de la gráfica dados por PLIM. El punto inicial Y0 especifica la línea a lo largo de la gráfica a partir de la cual

PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup] especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal.

FUN es el nombre del archivo con extensión M donde se encuentra definida la función del sistema continuo. Ver el archivo FUN.M

GRADFUN es el nombre del archivo con extensión M donde se define el Jacobiano de la función. Ver el archivo GRADFUN.M

PMUL(PLIM,'FUN','GRADFUN',Y0,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

PMUL(PLIM,'FUN','GRADFUN',Y0,PPLT,PPAR)

permite identificar las posiciones de los parámetros independientes `[a1,a2,...]` `PPAR` es un vector columna que contiene sus posiciones. Si `PPAR` no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

`PMUL(PLIM,'FUN','GRADFUN',Y0,PPLT,PPAR,S)`
permite especificas las distancias para calcular para determinar los puntos a partir de los cuales se determinarán los equilibrios (puntos fijos). `S` es un vector de dos o tres dimensiones cuyos componentes indican las distancias sobre cada uno de los ejes.

`PMUL(PLIM,'FUN','GRADFUN',Y0,PPLT,PPAR,S,DELTA,EPSILON)`
permite especificar la tolerancia para localizar las soluciones de `0=FUN(y)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son `10^(-5)`.

`PMUL(PLIM,'FUN','GRADFUN',Y0,PPLT,PPAR,S,DELTA,EPSILON,PREC)`
permite especificar la precisión para la presentación de los equilibrios encontrados. Por defecto `PREC=4`.

4.6.4 Ejemplo

Es posible detecta una bifurcación de codimensión 2 tal como la bifurcación *fork* para sistemas dinámicos continuos mediante la función `PMUL`. Si se define la forma normal de esta bifurcación junto con su Jacobiano en los archivos `FORK.M` y `FORKJ.M`, así:

```
function yp=fork(y)
x=y(1); alfa=y(2);
yp=alfa*x-x^3;
```

```
function [J,ValPr,Z]=ForkJ(y)
x=y(1); alfa=y(2);
J=alfa-3*x^2;
ValPr = eig(J);Z=x;
```

Gráficamente se puede localizar esta bifurcación del siguiente modo:

```
pmul([-1 1;-1 1],'Fork','ForkJ',[-1 -1],[2 1],2,[0.01 0.1])
```

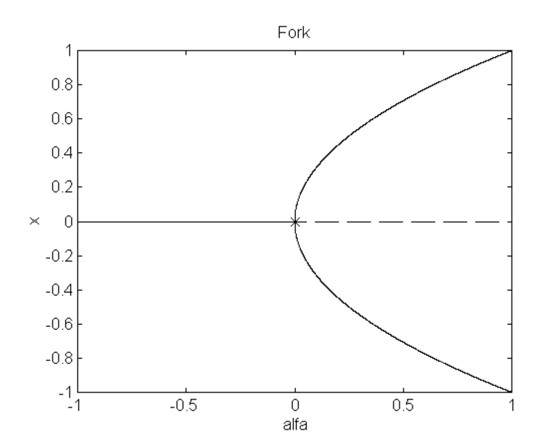


Figura 6 – Función PMUL en el IGB

Los equilibrios estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el equilibrio es indeterminado se grafica en gris (cruz).

4.6.5 Algoritmo

Utiliza el algoritmo del método de Newton – Raphson para la localización de los equilibrios. Comprueba la estabilidad evaluando el valor propio del punto fijo localizado.

4.6.6 Ver también

`PUMULD, FUN, GRADFUN, NEWTONMP`

4.6.7 Referencias

Capítulos 5, 6 y 7.

4.7 PMulD

4.7.1 Propósito

PMULD Identificación de Bifurcaciones a Fuerza Bruta mediante múltiples puntos de aproximación y solución con el Método de Newton-Raphson.

4.7.2 Sintaxis

```
function flag_stop=pmuld(PLIM,FUN,GRADFUN,GRADFUND,y0,PPLT,PPAR,...
                       s,DELTA,EPSILON,PREC)
```

4.7.3 Descripción

PMULD(PLIMS,'FUN','GRADFUN','GRADFUND',Y0) realiza el proceso de localización a partir de puntos definidos dentro de los límites de la gráfica dados por PLIM.

PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup] especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal.

FUN es el nombre del archivo con extensión M donde se encuentra definida la función del sistema continuo. Ver el archivo FUN.M

GRADFUN es el nombre del archivo con extensión M donde se define el Jacobiano de la función. Ver el archivo GRADFUN.M

GRADFUND es el nombre del archivo con extensión M donde se define el Jacobiano de la función discreta. Ver el archivo GRADFUND.M

PMULD(PLIM,'FUN','GRADFUN','GRADFUND',Y0,PPLT)

permite identificar las posiciones de los parámetros que se desea plotear. `PPLT` es un vector que contiene las posiciones de estas. Si `PPLT` no es dado, se plotearán las primeras variables.

`PMULD(PLIM,'FUN','GRADFUN','GRADFUND',Y0,PPLT,PPAR)`
permite identificar las posiciones de los parámetros independientes `[a1,a2,...]` `PPAR` es un vector columna que contiene sus posiciones. Si `PPAR` no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

`PMULD(PLIM,'FUN','GRADFUN','GRADFUND',Y0,PPLT,PPAR,S)`
permite especificas las distancias para calcular para determinar los puntos a partir de los cuales se determinarán los equilibrios (puntos fijos). S es un vector de dos o tres dimensiones cuyos componentes indican las distancias sobre cada uno de los ejes.

`PMULD(PLIMS,'FUN','GRADFUN','GRADFUND',Y0,PPLT,PPAR,S,DELTA, EPSILON)` permite especificar la tolerancia para localizar las soluciones de `0=FUN(y)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son `10^(-5)`.

4.7.4 Ejemplo

Es posible detectar una bifurcación de codimensión dos o de doblamiento de periodo tal como la bifurcación *flip* para sistemas dinámicos discretos mediante la función `PMULD`. Si se define el Jacobiano de la forma normal de esta bifurcación en el archivo `FLIPJD.M`. Así mismo se definen esta forma normal como función continua junto con su Jacobiano en los archivos `FLIP.M` y `FLIPJ.M`, de esta forma:

```
function
Yp=Flip(y)
X=y(1);alfa=y(2);
Yp=-(2+alfa)*x +
x^3;
```

```
function
[J,ValPr,Z]=FlipJ(y)
x=y(1);alfa=y(2);
J=-(2+alfa)+3*x^2;
ValPr=eig(J);Z=-x;
```

```
function
[J,ValPr]=FlipJD(y)
x=y(1); alfa=y(2);
J=-(1+alfa)+3*x^2;
ValPr=eig(J);
```

Para detectar la bifurcación se ejecuta `PMULD` así:

```
pmuld([-3 -1;-1 1],'Flip','FlipJ','FlipJD',[0 -3],[2 1],2,...
[0.01 0.1])
```

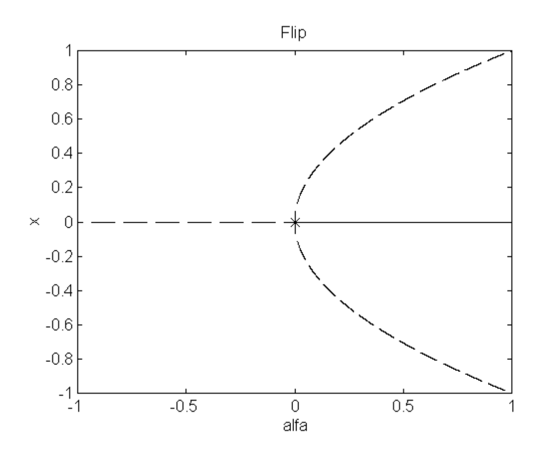


Figura 7 – Función PMULD en el IGB

Los puntos fijos estables se grafican en azul (línea continua) y los inestables en rojo (línea roja). Cuando el punto fijo es indeterminado se grafica en gris (cruz).

4.7.5 Algoritmo

Utiliza el algoritmo del método de Newton – Raphson para la localización de los equilibrios. Comprueba la estabilidad evaluando el valor propio del punto fijo localizado.

4.7.6 Ver también

`PUMUL, FUN, GRADFUN, GRADFUND, NEWTONMP`

4.7.7 Referencias

Capítulos 5, 6 y 7 .

4.8 PSecant

4.8.1 Propósito

PSECANT Identificación de bifurcaciones mediante la localización de equilibrios a lo largo de un ramal por continuación con corrector basado en el método secante.

4.8.2 Sintaxis

```
function flag_stop=psecant(PLIM,FUN,GRADFUN,TFUN,y0,PPLT,PPAR,s,...
                           DELTA,EPSILON,PREC)
```

4.8.3 Descripción

PSECANT(PLIM,'FUN','GRADFUN','TFUN',Y) realiza el proceso de continuación para la identificación de bifurcaciones en sistemas dinámicos continuos por el método de continuación con corrector basado en el método Secante, a partir del punto inicial Y.

PLIM especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup].

FUN es el nombre del archivo con extensión M donde se encuentra definida la función del sistema continuo. Ver el archivo FUN.M

GRADFUN es el nombre del archivo con extensión M donde se define el Jacobiano de la función. Ver el archivo GRADFUN.M

PSECANT(PLIM,'FUN','GRADFUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. Si PPLT no es dado, se plotearán las primeras variables.

PSECANT(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...]. PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

PSECANT(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S) permite especificas las distancias para calcular para determinar los puntos a partir de los cuales se determinarán los equilibrios (puntos fijos). S es un vector de dos o tres dimensiones cuyos componentes indican las distancias sobre cada uno de los ejes. Cuando S no está dado, se toma por defecto una fracción de 1/50 de los límites de la gráfica.

PSECANT(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

PSECANT(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON, PREC)permite especificar la precisión para la presentación de los puntos fijos encontrados. Por defecto PREC=4.

Método recomendado solo para sistemas dinámicos 2D.

4.8.4 Ejemplo

Si se define en los archivos FOLD y FOLDJ la forma normal de la bifurcación *fold* y su Jacobiano así,

```
function yp=fold(y)
x=y(1); alfa=y(2);
yp=alfa + x^2;
```

```
function [J,ValPr,Z]=foldj(y)
x=y(1); alfa=y(2);
J=2*x; Z=1;
ValPr=eig(J);
```

Es posible visualizar la bifurcación *fold* mediante la función PSECANT del siguiente modo,

```
figure; hold on
psecant([-1 1;-1 1],'Fold','FoldJ','TFun',[1 -1],[2 1],2,0.01)
psecant([-1 1;-1 1],'Fold','FoldJ','TFun',[1 -1],[2 1],2,0.01)
```

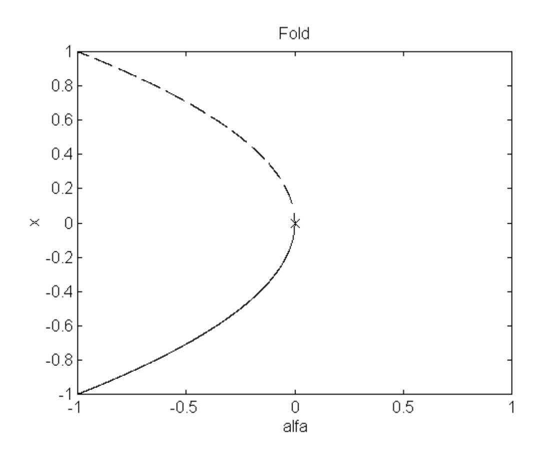


Figura 8 – Función PSECANT en el IGB

El archivo `TFUN.M` contiene la definición de la función de prueba. Los equilibrios estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el equilibrio es indeterminado se grafica en gris (cruz).

4.8.5 Algoritmo

Tal como se describe en el capítulo 5 , el algoritmo utilizado es:

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

4.8.6 Ver también

`PSECANTD, FUN, GRADFUN, TFUN`

4.8.7 Referencias

Capítulo 5.

4.9 PSecantD

4.9.1 Propósito

PSECANTD Identificación de bifurcaciones mediante la localización de puntos fijos a lo largo de un ramal por continuación con corrector basado en el método secante.

4.9.2 Sintaxis

```
function flag_stop=psecantd(PLIM,FUN,GRADFUND,TFUN,y0,PPLT,PPAR,...
                        s,DELTA,EPSILON,PREC)
```

4.9.3 Descripción

PSECANTD(PLIMS,'FUN','GRADFUND','TFUN',Y) realiza el proceso de continuación para la identificación de bifurcaciones en sistemas dinámicos discretos por el método de continuación con corrector basado en el método Secante, a partir del punto inicial Y.

PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup] especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal.

FUN es el nombre del archivo con extensión M donde se encuentra definida la función del sistema continuo. Ver el archivo FUN.M

GRADFUND es el nombre del archivo con extensión M donde se define el Jacobiano de la función discreta. Ver el archivo GRADFUND.M

TFUN es el nombre del archivo con extensión M donde se encuentra definida la función del sistema continuo. Ver el archivo FUN.M

PSECANTD(PLIM,'FUN','GRADFUND','TFUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. Si PPLT no es dado, se plotearán las primeras variables.

PSECANTD(PLIM,'FUN','GRADFUND','TFUN',Y,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...]. PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

PSECANTD(PLIM,'FUN','GRADFUND','TFUN',Y,PPLT,PPAR,S) permite especificas las distancias para calcular para determinar los puntos a partir de los cuales se determinarán los equilibrios (puntos fijos). S es un vector de dos o tres dimensiones cuyos componentes indican las distancias sobre cada uno de los ejes. Cuando S no está dado, se toma por defecto una fracción de 1/50 de los límites de la gráfica.

PSECANTD(PLIM,'FUN','GRADFUND','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

PSECANTD(PLIM,'FUN','GRADFUND','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON, PREC) permite especificar la precisión para la presentación de los puntos fijos encontrados. Por defecto PREC=4.

Método recomendado solo para sistemas dinámicos 2D.

4.9.4 Ejemplo

Es posible detectar una bifurcación *flip* para sistemas dinámicos discretos mediante la función PSECANTD. Si se define el Jacobiano de la forma normal de esta bifurcación en el archivo FLIPJD.M. Así mismo se definen esta forma normal como función continua en el archivo FLIP.M, de esta forma:

```
Function
yp=Flip(y)
```

```
function [J,ValPr]=FlipJD(y)
x=y(1); alfa=y(2);
```

```
x=y(1);alfa=y(2);                J=-(1+alfa)+3*x^2;
yp=-(2+alfa)*x + x^3;            ValPr=eig(J);
```

Si la función `PSECATND` se ejecuta del siguiente modo, se obtiene:

```
psecantd([-3 -1;-1 1],'Flip','FlipJD','TFunD',[0 -3],[2 1],2,0.01)
```

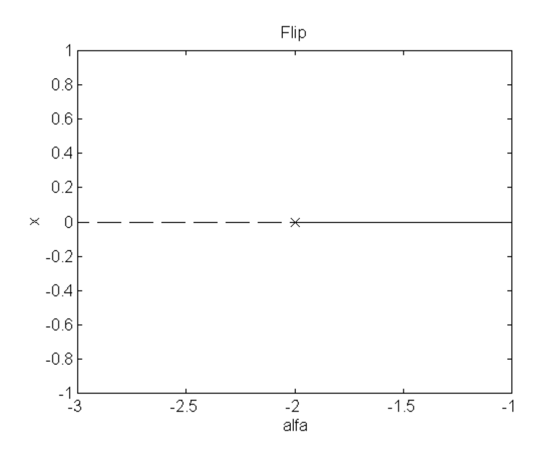


Figura 9 – Función PSACANTD en el IGB

El archivo `TFUND.M` contiene la definición de la función de prueba. Los puntos fijos estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el punto fijo es indeterminado se grafica en gris (cruz).

4.9.5 Algoritmo

Tal como se describe en el capítulo 5 , el algoritmo utilizado es:

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

4.9.6 Ver también

`PSECANT, FUN, GRADFUN, TFUND`

4.9.7 Referencias

Capítulos 5 y 7.

4.10 PSeidel

4.10.1 Propósito

PSEIDEL Identificación de bifurcaciones mediante la localización de puntos fijos a lo largo de un ramal por continuación con corrector basado en el método de Seidel para sistemas dinámicos discretos.

4.10.2 Sintaxis

```
function flag_stop=pseidel(PLIM,FUND,GRADFUND,TFUN,y0,PPLT,PPAR,...
                            s,TOL,PREC)
```

4.10.3 Descripción

PSEIDEL(PLIM,'FUND','GRADFUND','TFUN',Y)
realiza el proceso de continuación para la identificación de bifurcaciones en sistemas dinámicos discretos por el método de continuación con corrector basado en el método Secante, a partir del punto inicial Y.

PLIM especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. PLIM es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup].

FUND es el nombre del archivo con extensión M donde se encuentra definida la función del sistema discreto. Ver el archivo FUND.M

GRADFUND es el nombre del archivo con extensión M donde se encuentra definido el Jacobiano de la función del sistema discreto. Ver el archivo GRADFUND.M

TFUN es el nombre del archivo con extensión M donde se encuentra definida la función de prueba. Ver el archivo TFUND.M

PSEIDEL(PLIM,'FUND','GRADFUND','TFUN',Y,PPLT)

permite identificar las posiciones de los parámetros que se desea plotear. `PPLT` es un vector que contiene las posiciones de estas. Si `PPLT` no es dado, se plotearán las primeras variables.

`PSEIDEL(PLIM,'FUND','GRADFUND','TFUN',Y,PPLT,PPAR)`
permite identificar las posiciones de los parámetros independientes `[a1,a2,...]`. `PPAR` es un vector columna que contiene sus posiciones. Si `PPAR` no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

`PSEIDEL(PLIM,'FUND','GRADFUND','TFUN',Y,PPLT,PPAR,S)`
permite especificas las distancias para calcular para determinar los puntos a partir de los cuales se determinarán los equilibrios (puntos fijos). `S` es un vector de dos o tres dimensiones cuyos componentes indican las distancias sobre cada uno de los ejes.

`PSEIDEL(PLIM,'FUND','GRADFUND','TFUN',Y,PPLT,PPAR,S,TOL)`
permite especificar la tolerancia para localizar las soluciones de `0=FUND(y0)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son `10^(-5)`.

`PSEIDEL(PLIM,'FUND',Y,'GRADFUND','TFUN',PPLT,PPAR,S,TOL,PREC)` permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

Solo puede detectar puntos fijos estables en sistemas dinámicos discretos, por lo que la definición del archivo donde se define el Jacobiano y de la función de prueba pueden omitirse.

4.10.4 Ejemplo

En el caso de la forma normal de la bifurcación *flip* para sistemas dinámicos discretos, es posible detectar los puntos fijos estables de la función discreta definida en el archivo `FLIPD.M`. La estabilidad puede comprobarse evaluando los valores propios en el archivo `FLIPJD.M`, definidos así:

```
function yp=FlipD(y)
x=y(1); alfa=y(2);
yp=-(1+alfa)*x + x^3;
```

```
function [J,ValPr]=FlipJD(y)
x=y(1); alfa=y(2);
J=-(1+alfa)+3*x^2;
ValPr = eig(J);
```

Si la función `PSEIDEL` se ejecuta del siguiente modo, se obtiene:

```
pseidel([-3 0;-1 1],'FlipD','FlipJD','TFunD',[0 -1.9],...
                [2 1],2,0.01,10^(-5))
```

Figura 10 – Función PSEIDEL en el IGB

Los puntos fijos estables se grafican en azul (línea continua). El archivo `TFUND.M` contiene la definición de la función de prueba. Este archivo puede omitirse ya que no hay cambio en la estabilidad de los puntos fijos detectados. Lo mismo con el archivo del Jacobiano, ya que todos los puntos fijos hallados serán estables.

4.10.5 Algoritmo

Se utiliza el algoritmo del método de Seidel, es decir:

```
fdX=feval(fname,X);
for i=1:length(fdX)
      Xp(i)=fdX(i);
      fdX=feval(fname,Xp);
end
X=Xp;
```

4.10.6 Ver también

`SEIDELMP`

4.10.7 Referencias

Capítulo 5.

5. Métodos de continuación

El IGB identifica los valores críticos de los parámetros independientes de un sistema dinámico continuo o discreto, por medio de funciones para la localización de equilibrios y puntos fijos a lo largo de ramales. Estas técnicas se basan en los métodos de continuación predictor – corrector. El corrector utilizado en todos estos métodos se basa en el método de Newton – Raphson. En cuanto al predictor, se utilizan dos métodos: predictor secante y predictor tangente, cada uno con y sin parametrización.

Las funciones que implementan los métodos de continuación con predictor secante son:

Sin parametrización para sistemas continuos y discretos:

- CSC
- CSCD

Parametrizado para sistemas continuos y discretos

- CSCP
- CSCPD

Las funciones que implementan los métodos de continuación con predictor tangente son:

Sin parametrización para sistemas continuos y discretos

- CTC
- CTCD

Parametrizado para sistemas continuos y discretos

- CTCP
- CTCPD

5.1 CSC

5.1.1 Propósito

Localizar bifurcaciones por continuación con predictor secante sin parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos continuos.

5.1.2 Sintaxis

```
flag_stop=csc(PLIM,FUN,GRADFUN,TFUN,y0,PPLT,PPAR,s,...
                   DELTA,EPSILON,PREC)
```

5.1.3 Descripción

`CSC(PLIM,'FUN','GRADFUN','TFUN',Y)` realiza el proceso de continuación secante con parametrización por longitud de arco de la función descrita en el archivo `'FUN'` a partir de los puntos iniciales `Y1=Y` y `Y2=Y+S`, donde `S` es por defecto es una centésima parte de la longitud del eje horizontal.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`.

`'GRADFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`.

`'TFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función de prueba. Ver el archivo `'TFUN.M'`.

CSC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

CSC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...]. PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

CSC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es un vector de la misma dimensión de Y0, que indica el tamaño del primer paso. Su signo indica la dirección de este. Si S no es dado, se asume como de una centésima del eje horizontal.

CSC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(Y), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

CSC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON,PREC) permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

Para la detección de ciertas bifurcaciones se hace necesario utilizar un tamaño de paso muy pequeño para lograr la localización de un equilibrio al otro lado de la bifurcación.

5.1.4 Ejemplo

La función,

$$\alpha + x^2 = 0$$

que describe la forma normal de la bifurcación de *fold*, se encuentra descrita en el archivo FOLD.M, así

```
function yp=fold(y)
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de esta función se encuentra definido en el archivo `FOLDJ.M`, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La bifurcación de *fold* se puede localizar en este sistema mediante la función `CSC`, así:

```
csc([-1 1;-1 1],'Fold','FoldJ','TFun',[-1 -1],[2 1],2,0.0001)
```

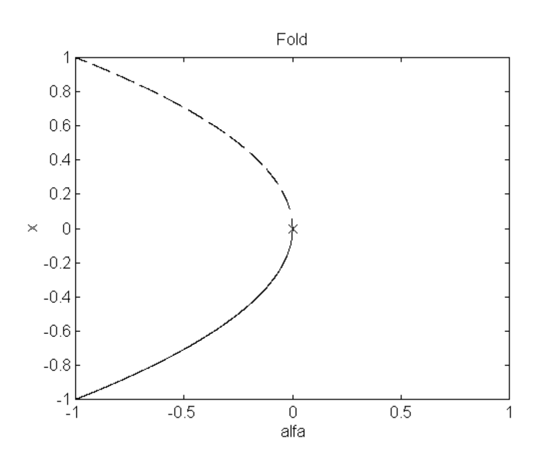


Figura 11 – Función CSC en el IGB

Los equilibrios estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el equilibrio es indeterminado se grafica en gris (cruz). En el archivo `TFUN.M` se encuentra definida la función de prueba que permite identificar la bifurcación directamente en medio de dos equilibrios. Este resultado se consigna en el archivo `RESULTADOS.TXT`.

```
type resultados.txt

IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
8-May-0
```

```
Método: Predictor Secante Continuo
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo de la función de prueba: TFun

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación FOLD entre los equilibrios
Y=[-0.001412; -1.993e-006] e-valor=-0.002823
Y=[0.002769; -7.667e-006] e-valor=0.005538
```

5.1.5 Algoritmo

Los valores iniciales de los dos puntos necesarios para efectuar el método de continuación con predictor secante, se calculan con el método de Newton – Raphson multiparamétrico del siguiente modo,

```
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
y2=y1+s;
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción secante es,

```
h=norm(y2-y1);
s=(y2-y1)/norm(y2-y1);
y1=y2; y2=y2+h*s; eig1=eig2;
```

Finalmente, la corrección se hace utilizando el método de Newton – Raphson multimparamétrico,

```
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.1.6 Ver también

```
CSCD, CSCP, CSCPD
```

5.1.7 Referencias

Capítulo 6 .

5.2 CSCD

5.2.1 Propósito

Localizar bifurcaciones por continuación con predictor secante sin parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos discretos.

5.2.2 Sintaxis

```
flag_stop=cscd(PLIM,FUN,GRADFUN,GRADFUND,TFUN,y0,...
           PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

5.2.3 Descripción

`CSCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y)` realiza el proceso de continuación secante con parametrización por longitud de arco de la función descrita en el archivo `'FUN'` a partir de los puntos iniciales `Y1=Y` y `Y2=Y+S`, donde `S` es por defecto es una centésima parte de la longitud del eje horizontal.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión .M donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`

`'GRADFUN'` es el nombre del archivo con extensión .M donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`

`'GRADFUND'` es el nombre del archivo con extensión .M donde se encuentra definido el Jacobiano de la función del sistema discreto. Ver el archivo `'GRADFUND.M'`

`'TFUN'` es el nombre del archivo con extensión .M donde se encuentra definida la función de prueba. Ver el archivo `'TFUND.M'`

`CSCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT)` permite identificar las posiciones de los parámetros que se desea plotear. `PPLT` es un vector que contiene las posiciones de estas. Si `PPLT` no es dado, se plotearán las primeras variables.

`CSCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT,PPAR)`
permite identificar las posiciones de los parámetros independientes `[a1,a2,...]`. `PPAR` es un vector columna que contiene sus posiciones. Si `PPAR` no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

`CSCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT,PPAR,S)`
permite especificar el tamaño y la dirección del paso inicial. `S` es un vector de la misma dimensión de `Y0`, que indica el tamaño del primer paso. Su signo indica la dirección de este. Si `S` no es dado, se asume como de una centésima del eje horizontal.

`CSCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON)` Permite especificar la tolerancia para localizar las soluciones de `0=FUN(y0)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son `10^(-5)`.

`CSCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON,PREC)` permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

Para la detección de ciertas bifurcaciones se hace necesario utilizar un tamaño de paso muy pequeño para lograr la localización de un equilibrio al otro lado de la bifurcación.

5.2.4 Ejemplo

La función,

$$x \mapsto \alpha + x + x^2$$

que describe la forma normal de la bifurcación de *fold* en un sistema dinámico discreto se encuentra descrita en el archivo FOLDD.M, así

```
function yp=foldd(y)
x=y(1); alfa=y(2);
yp=alfa+x+x^2;
```

La función equivalente como sistema dinámico continuo, se encuentra descrita en el archivo FOLD.M, así

```
function yp=fold(y)
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de la función discreta se encuentra definido en el archivo FOLDJD.M, así

```
function [J,ValPr]=foldjd(y)
x=y(1); alfa=y(2);
J=1+2*x; ValPr=eig(J);
```

El Jacobiano de la función continua se encuentra definido en el archivo FOLDJ.M, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La bifurcación de *fold* en este sistema dinámicos discreto se puede localizar en este sistema mediante la función CSCD, así:

```
cscd([-1 1;-1 1],'Fold','FoldJ','FoldJd','TFunD',[-1 -1],...
[2 1],2,0.0001)
```

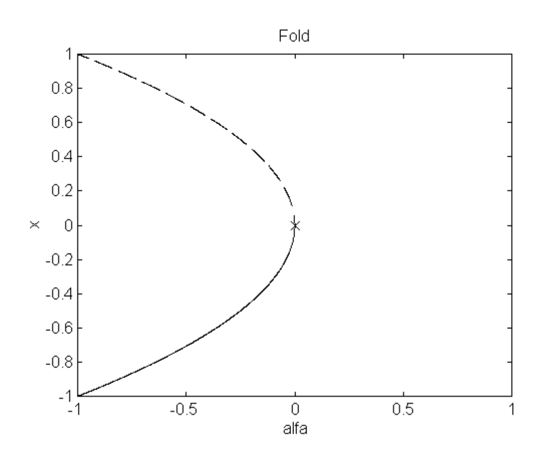


Figura 12 – Función CSCD en el IGB

Los puntos fijos estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el punto fijo es indeterminado se grafica en gris (cruz). En el archivo `TFUND.M` se encuentra definida la función de prueba que permite identificar la bifurcación directamente en medio de dos puntos fijos. Este resultado se consigna en el archivo `RESULTADOS.TXT`.

```
type resultados.txt

IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
8-May-0

Método: Predictor Secante Discreto
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo del Jacobiano: FoldJd
Archivo de la función de prueba: TFunD

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación DESCONOCIDA entre
Y=[-1; -1] e-valor=-1
Y=[-0.9999; -0.9999] e-valor=-0.9999

Bifurcación FOLD entre los puntos fijos
Y=[-0.001412; -1.993e-006] e-valor=0.9972
Y=[0.002769; -7.667e-006] e-valor=1.006
```

5.2.5 Algoritmo

Los valores iniciales de los dos puntos necesarios para efectuar el método de continuación con predictor secante, se calculan con el método de Newton – Raphson multiparamétrico del siguiente modo,

```
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
y2=y1+s;
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción secante es,

```
h=norm(y2-y1);
s=(y2-y1)/norm(y2-y1);
y1=y2; y2=y2+h*s; eig1=eig2;
```

Finalmente la corrección se hace utilizando el método de Newton – Raphson multimparamétrico,

```
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.2.6 Ver también

`CSC, CSCP, CSCPD`

5.2.7 Referencias

Capítulo 6.

5.3 CSCP

5.3.1 Propósito

Localizar bifurcaciones por continuación con predictor secante con parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos continuos.

5.3.2 Sintaxis

```
flag_stop=cscp(PLIM,FUN,GRADFUN,FUNP,GRADFUNP,TFUN,y0,...
                    PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

5.3.3 Descripción

`CSCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y)` realiza el proceso de continuación secante con parametrización por longitud de arco de la función descrita en el archivo `'FUN'` a partir de los puntos iniciales `Y1=Y` y `Y2=Y+S`, donde `S` es por defecto es una centésima parte de la longitud del eje horizontal.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`

`'GRADFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`

`'FUNP'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo parametrizada. Ver el archivo `'FUNP.M'`

'TFUN' es el nombre del archivo con extensión .M donde se encuentra definida la función de prueba. Ver el archivo 'TFUN.M'

CSCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

CSCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR) Permite identificar las posiciones de los parámetros independientes [a1,a2,...]. PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

CSCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es el tamaño de paso (longitud de arco). Su signo indica hacia donde se da el primer paso. Si S no es dado, se asume como de una centésima del eje horizontal.

CSCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

CSCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON,PREC) permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

5.3.4 Ejemplo

La función,

$$\alpha + x^2 = 0$$

que describe la forma normal de la bifurcación de *fold*, se encuentra descrita en el archivo FOLD.M, así

```
function yp=fold(y)
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de esta función se encuentra definido en el archivo `FOLDJ.M`, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La función continua parametrizada respecto a la longitud de arco se describe en el archivo `FOLDP.M`, así

```
function ypp=foldp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
ypp=[alfa + x^2;(x-x0)^2 + (alfa-alfa0)^2 - h^2];
```

El Jacobiano de esta función continua parametrizada, se describe en el archivo `FOLDJP.M`, así

```
function Jp=foldjp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
J=foldj(y);Z=1;
dP=[2*(x-x0),2*(alfa-alfa0)];
Jp=[J,Z;dP];
```

La bifurcación de *fold* se puede localizar en este sistema mediante la función `CSCP`, así:

```
cscp([-1 1;-1 1],'Fold','FoldJ','FoldP','FoldJP','TFun',...
[-1 -1],[2 1],2,0.001)
```

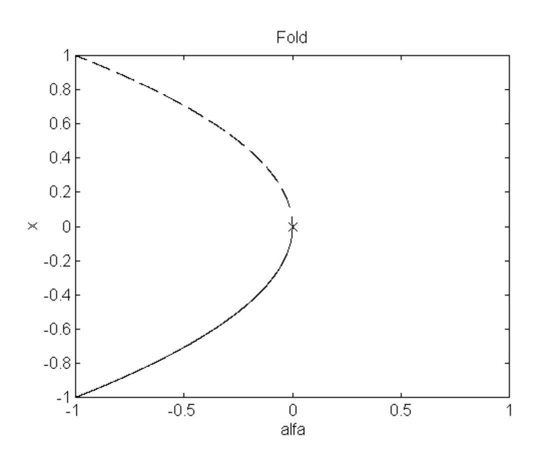


Figura 13 – Función CSCP en el IGB

Los equilibrios estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el equilibrio fijo es indeterminado se grafica en gris (cruz). En el archivo `TFUN.M` se encuentra definida la función de prueba que permite identificar la bifurcación directamente en medio de dos puntos fijos. Este resultado se consigna en el archivo `RESULTADOS.TXT`.

```
type resultados.txt

IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
8-May-0

Método: Predictor Secante Continuo con Parametrización
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo del Jacobiano parametrizado: FoldJP
Archivo de la función de prueba: TFun

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación FOLD entre los equilibrios
Y=[-0.01381; -0.0001906] e-valor=-0.02761
Y=[0.0003355; -1.125e-007] e-valor=0.000671
```

5.3.5 Algoritmo

Los valores iniciales de los dos puntos necesarios para efectuar el método de continuación con predictor secante, se calculan con el método de Newton – Raphson multiparamétrico del siguiente modo,

```
MPPAR=length(y0)+1:2*length(y0)+1;
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
y2=y1+s;
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción secante es,

```
h=norm(y2-y1);
s=(y2-y1)/norm(y2-y1);
y1=y2; y2=y2+h*s; eig1=eig2;
```

Finalmente la corrección se hace utilizando el método de Newton – Raphson multimparamétrico, con base en la función parametrizada y su Jacobiano, así

```
y2=newtonmp(FUNP,GRADFUNP,[y2',y1',h],MPPAR',DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.3.6 Ver también

`CSC, CSCD, CSCPD`

5.3.7 Referencias

Capítulo 6 .

5.4 CSCPD

5.4.1 Propósito

Localizar bifurcaciones por continuación con predictor secante con parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos discretos.

5.4.2 Sintaxis

```
flag_stop=cscpd(PLIM,FUN,GRADFUN,FUNP,GRADFUNP,...
 GRADFUND,TFUN,y0,PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

5.4.3 Descripción

`CSCPD(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y)` realiza el proceso de continuación secante con parametrización por longitud de arco de la función descrita en el archivo `'FUN'` a partir de los puntos iniciales `Y1=Y` y `Y2=Y+S`, donde `S` es por defecto es una centésima parte de la longitud del eje horizontal.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`

`'GRADFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`

`'FUNP'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo parametrizada. Ver el archivo `'FUNP.M'`

'GRADFUNP' es el nombre del archivo con extensión .M donde se encuentra definido el Jacobiano de la función del sistema continuo parametrizado. Ver el archivo 'GRADFUNP.M'

'GRADFUND' es el nombre del archivo con extensión .M donde se encuentra definido el Jacobiano de la función del sistema discreto. Ver el archivo 'GRADFUND.M'

'TFUN' es el nombre del archivo con extensión .M donde se encuentra definida la función de prueba. Ver el archivo 'TFUND.M'

CSCPD(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

CSCPD(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...]. PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

CSCPD(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es el tamaño de paso (longitud de arco). Su signo indica hacia donde se da el primer paso. Si S no es dado, se asume como de una centésima del eje horizontal.

CSCPD(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

`CSCPD(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR,S,DELTA,EPSILON,PREC)` permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

5.4.4 Ejemplo

La función,

$$x \mapsto \alpha + x + x^2$$

que describe la forma normal de la bifurcación de *fold* en un sistema dinámico discreto se encuentra descrita en el archivo `FOLDD.M`, así

```
function yp=foldd(y)
x=y(1); alfa=y(2);
yp=alfa+x+x^2;
```

La función equivalente como sistema dinámico continuo, se encuentra descrita en el archivo `FOLD.M`, así

```
function yp=fold(y)
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de la función discreta se encuentra definido en el archivo `FOLDJD.M`, así

```
function [J,ValPr]=foldjd(y)
x=y(1); alfa=y(2);
J=1+2*x; ValPr=eig(J);
```

El Jacobiano de la función continua se encuentra definido en el archivo `FOLDJ.M`, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La función continua parametrizada respecto a la longitud de arco se describe en el archivo `FOLDP.M`, así

```
function ypp=foldp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
ypp=[alfa + x^2;(x-x0)^2 + (alfa-alfa0)^2 - h^2];
```

El Jacobiano de esta función continua parametrizada, se describe en el archivo FOLDJP.M, así

```
function Jp=foldjp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
J=foldj(y);Z=1;
dP=[2*(x-x0),2*(alfa-alfa0)];
Jp=[J,Z;dP];
```

La bifurcación de *fold* se puede localizar en este sistema mediante la función CSCPD, así:

```
cscpd([-1 1;-1 1],'Fold','FoldJ','FoldP','FoldJP','FoldJD',...
'TFun',[-1 -1],[2 1],2,0.001)
```

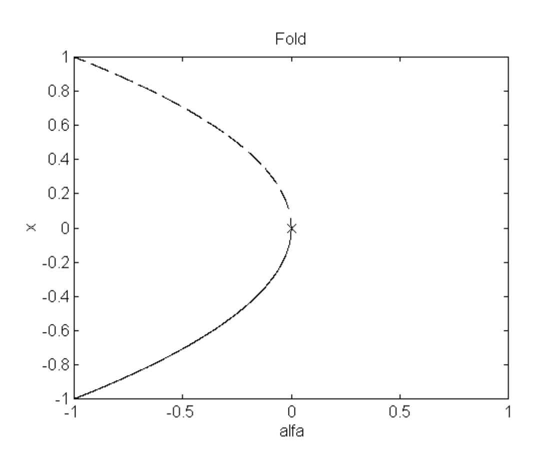


Figura 14 – Función CSCPD en el IGB

Los puntos fijos estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el punto fijo es indeterminado se grafica en gris (cruz). En el archivo TFUND.M se encuentra definida la función de prueba que permite identificar la bifurcación directamente en medio de dos puntos fijos. Este resultado se consigna en el archivo RESULTADOS.TXT.

```
type resultados.txt

IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
8-May-0
```

```
Método: Predictor Secante Discreto con Parametrización
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo de la función parametrizada: FoldP
Archivo del Jacobiano parametrizado: FoldJP
Archivo de la función de prueba: TFun

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación FOLD entre los equilibrios
Y=[-0.5; -0.3] e-valor=-0.008
Y=[-0.5; -0.2] e-valor=0.01
```

5.4.5 Algoritmo

Los valores iniciales de los dos puntos necesarios para efectuar el método de continuación con predictor secante, se calculan con el método de Newton – Raphson multiparamétrico del siguiente modo,

```
MPPAR=length(y0)+1:2*length(y0)+1;
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
y2=y1+s;
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción secante es,

```
h=norm(y2-y1);
s=(y2-y1)/norm(y2-y1);
y1=y2; y2=y2+h*s; eig1=eig2;
```

Finalmente la corrección se hace utilizando el método de Newton – Raphson multimparamétrico, con base en la función parametrizada y su Jacobiano, así

```
y2=newtonmp(FUNP,GRADFUNP,[y2',y1',h],MPPAR',DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.4.6 Ver también

CSC, CSCD, CSCP

5.4.7 Referencias

Capítulo 6 .

5.5 CTC

5.5.1 Propósito

Localizar bifurcaciones por continuación con predictor tangente sin parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos continuos.

5.5.2 Sintaxis

```
flag_stop=ctc(PLIM,FUN,GRADFUN,TFUN,y0,...
     PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

5.5.3 Descripción

`CTC(PLIM,'FUN','GRADFUN','TFUN',Y)` realiza el proceso de continuación tangente con parametrización por longitud de arco de la función descrita en el archivo `'FUN'` a partir de un punto inicial `Y=[y1,y2,...]` calculando su derivada descrita en la función `'GRADFUN'`.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`

`'GRADFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`

`'TFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función de prueba. Ver el archivo `'TFUN.M'`

`CTC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT)` permite identificar las posiciones de los parámetros que se desea plotear. `PPLT` es un vector que contiene las posiciones de estas. Si `PPLT` no es dado, se plotearán las primeras variables.

`CTC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR)` permite identificar las posiciones de los parámetros independientes `[a1,a2,...]`. `PPAR` es un vector columna que contiene sus posiciones. Si `PPAR` no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

`CTC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S)` permite especificar el tamaño y la dirección del paso inicial. `S` es un vector de la misma dimensión de `Y0`, que indica el tamaño del primer paso. Su signo indica la dirección de este. Si `S` no es dado, se asume como de una centésima del eje horizontal.

`CTC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON)` permite especificar la tolerancia para localizar las soluciones de `0=FUN(y0)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son `10^(-5)`.

`CTC(PLIM,'FUN','GRADFUN','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON,PREC)` permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

Con este método no se puede sobrepasar la bifurcación de *fold*, y en general ningún ramal que tenga una recta tangente con pendiente infinita.

5.5.4 Ejemplo

La función,

$$\alpha + x^2 = 0$$

que describe la forma normal de la bifurcación de *fold*, se encuentra descrita en el archivo `FOLD.M`, así

```
function yp=fold(y)
```

```
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de esta función se encuentra definido en el archivo `FOLDJ.M`, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La bifurcación de *fold* se puede localizar en este sistema mediante la función `CTC`, así:

```
ctc([-1 1;-1 1],'Fold','FoldJ','TFun',[-1 -1],[2 1],2,0.0001)
ctc([-1 1;-1 1],'Fold','FoldJ','TFun',[1 -1],[2 1],2,0.0001)
```

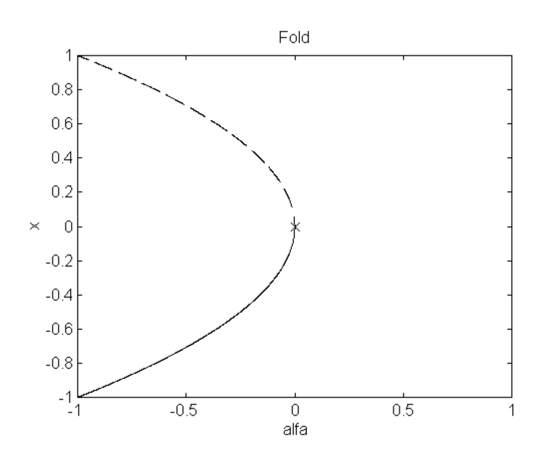


Figura 15 – Función CTC en el IGB

Para la identificación gráfica de esta bifurcación en particular, la función `CTC` debe ser ejecutada dos veces tal como se aprecia, ya que con este método no se puede rebasar el punto de bifurcación.

Los equilibrios estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el equilibrio es indeterminado se grafica en gris (cruz). En el archivo `TFUN.M` se encuentra definida la función de prueba que permite identificar la bifurcación directamente en medio de dos equilibrios. Este resultado se consigna en el archivo `RESULTADOS.TXT`.

```
type resultados.txt
```

```
IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
9-May-0

Método: Predictor Tangente Continuo
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo de la función de prueba: TFun

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación FOLD entre los equilibrios
Y=[-0.0002951; -8.707e-008] e-valor=-0.0005902
Y=[0.0002829; 1.216e-007] e-valor=0.0005658
```

5.5.5 Algoritmo

El primer punto fijo se calcula a partir del punto inicial `y0` mediante el método de Newton – Raphson de la siguiente forma,

```
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción tangente es,

```
e=y0*0;
e(length(y0))=1;
JZe=[J,Z;e'];
s=inv(JZe)*e;
s=s/norm(s);
y2=y1+h*s;
```

Finalmente, la corrección se hace utilizando el método de Newton – Raphson multiparamétrico,

```
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.5.6 Ver también

CTCD, CTCP, CTCPD

5.5.7 Referencias

Capítulo 6

5.6 CTCD

5.6.1 Propósito

Localizar bifurcaciones por continuación con predictor tangente sin parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos discretos.

5.6.2 Sintaxis

```
flag_stop=ctcd(PLIM,FUN,GRADFUN,GRADFUND,TFUN,y0,...
           PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

5.6.3 Descripción

`CTCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y)` realiza el proceso de continuación tangente con parametrización por longitud de arco de la función descrita en el archivo `'FUN'` a partir de un punto inicial `Y=[y1,y2,...]` calculando su derivada descrita en la función `'GRADFUN'`.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`

`'GRADFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`

`'GRADFUND'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema discreto. Ver el archivo `'GRADFUND.M'`

`'TFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función de prueba. Ver el archivo `'TFUND.M'`

`CTCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT)`
permite identificar las posiciones de los parámetros que se desea plotear. `PPLT` es un vector que contiene las posiciones de estas. Si `PPLT` no es dado, se plotearán las primeras variables.

`CTCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',,Y,PPLT,PPAR)`
permite identificar las posiciones de los parámetros independientes `[a1,a2,...]`. `PPAR` es un vector columna que contiene sus posiciones. Si `PPAR` no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

`CTCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT,PPAR,S)`
permite especificar el tamaño y la dirección del paso inicial. `S` es un vector de la misma dimensión de `Y0`, que indica el tamaño del primer paso. Su signo indica la dirección de este. Si `S` no es dado, se asume como de una centésima del eje horizontal.

`CTCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT,PPAR,S,DELTA, EPSILON)` permite especificar la tolerancia para localizar las soluciones de `0=FUN(y0)`, utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son `10^(-5)`.

`CTCD(PLIM,'FUN','GRADFUN','GRADFUND','TFUN',Y,PPLT,PPAR,S,DELTA, EPSILON,PREC)` permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

5.6.4 Ejemplo

La función,

$$x \mapsto \alpha + x + x^2$$

que describe la forma normal de la bifurcación de *fold* en un sistema dinámico discreto se encuentra descrita en el archivo `FOLDD.M`, así

```
function yp=foldd(y)
x=y(1); alfa=y(2);
yp=alfa+x+x^2;
```

La función equivalente como sistema dinámico continuo, se encuentra descrita en el archivo `FOLD.M`, así

```
function yp=fold(y)
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de la función discreta se encuentra definido en el archivo `FOLDJD.M`, así

```
function [J,ValPr]=foldjd(y)
x=y(1); alfa=y(2);
J=1+2*x; ValPr=eig(J);
```

El Jacobiano de la función continua se encuentra definido en el archivo `FOLDJ.M`, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La bifurcación de *fold* en este sistema dinámicos discreto se puede localizar en este sistema mediante la función `CTCD`, así:

```
ctcd([-1 1;-1 1],'Fold','FoldJ','FoldJd','TFunD',[-1 -1],...
[2 1],2,0.0001)
```

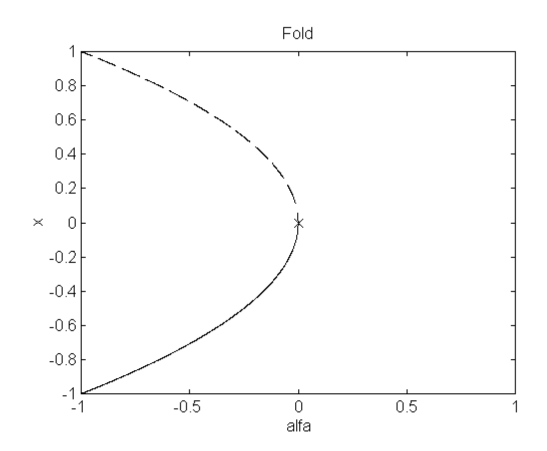


Figura 16 – Función CTCD en el IGB

Los puntos fijos estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el punto fijo es indeterminado se grafica en gris (cruz). En el archivo `TFUND.M` se encuentra definida la función de prueba que permite identificar la bifurcación directamente en medio de dos equilibrios. Este resultado se consigna en el archivo `RESULTADOS.TXT`.

```
type resultados.txt

IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
9-May-0

Método: Predictor Secante Discreto
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo del Jacobiano: FoldJd
Archivo de la función de prueba: TFunD

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación DESCONOCIDA entre
Y=[-1; -1] e-valor=-1
Y=[-0.9999; -0.9999] e-valor=-0.9999

Bifurcación FOLD entre los puntos fijos
Y=[-0.001412; -1.993e-006] e-valor=0.9972
Y=[0.002769; -7.667e-006] e-valor=1.006
```

5.6.5 Algoritmo

El primer punto fijo se calcula a partir del punto inicial `y0` mediante el método de Newton – Raphson de la siguiente forma,

```
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción tangente es,

```
e=y0*0;
e(length(y0))=1;
JZe=[J,Z;e'];
s=inv(JZe)*e;
s=s/norm(s);
```

```
y2=y1+h*s;
```

Finalmente, la corrección se hace utilizando el método de Newton – Raphson multimparamétrico,

```
y2=newtonmp(FUN,GRADFUN,y2,PPAR,DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.6.6 Ver también

CTC, CTCP, CTCPD

5.6.7 Referencias

Capítulo 6 .

5.7 CTCP

5.7.1 Propósito

Localizar bifurcaciones por continuación con predictor tangente con parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos continuos.

5.7.2 Sintaxis

```
flag_stop=ctcp(PLIM,FUN,GRADFUN,FUNP,GRADFUNP,TFUN,y0,...
          PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

5.7.3 Descripción

`CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y)` realiza el proceso de continuación tangente con parametrización por longitud de arco de la función descrita en el archivo `'FUN'` a partir de un punto inicial `Y=[y1,y2,...]` calculando su derivada descrita en la función `'GRADFUN'`.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`

`'GRADFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`

`'FUNP'` es el nombre del archivo con extensión .M donde se encuentra definida la función del sistema continuo parametrizada. Ver el archivo `'FUNP.M'`

'GRADFUNP' es el nombre del archivo con extensión .M donde se encuentra definido el Jacobiano de la función del sistema continuo parametrizado. Ver el archivo 'GRADFUNP.M'

'TFUN' es el nombre del archivo con extensión .M donde se encuentra definida la función de prueba. Ver el archivo 'TFUND.M'

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...] PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es el tamaño de paso (longitud de arco). Su signo indica hacia donde se da el primer paso. Si S no es dado, se asume como de una centésima del eje horizontal.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^(-5).

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','TFUN',Y,PPLT,PPAR,S,DELTA, EPSILON,PREC) permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

Con este método no se puede sobrepasar la bifurcación de *fold*, y en general ningún ramal que tenga una recta tangente con pendiente infinita. En este punto la función entra en un *loop*.

5.7.4 Ejemplo

La función,

$$\alpha + x^2 = 0$$

que describe la forma normal de la bifurcación de *fold*, se encuentra descrita en el archivo `FOLD.M`, así

```
function yp=fold(y)
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de esta función se encuentra definido en el archivo `FOLDJ.M`, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La función continua parametrizada respecto a la longitud de arco se describe en el archivo `FOLDP.M`, así

```
function ypp=foldp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
ypp=[alfa + x^2;(x-x0)^2 + (alfa-alfa0)^2 - h^2];
```

El Jacobiano de esta función continua parametrizada, se describe en el archivo `FOLDJP.M`, así

```
function Jp=foldjp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
J=foldj(y);Z=1;
dP=[2*(x-x0),2*(alfa-alfa0)];
Jp=[J,Z;dP];
```

La bifurcación de *fold* se puede localizar en este sistema mediante la función `CTCP`, así:

```
ctcp([-1 1;-1 1],'Fold','FoldJ','FoldP','FoldJP','TFun',...
[1 -1],[2 1],2,0.0001)
ctcp([-1 1;-1 1],'Fold','FoldJ','FoldP','FoldJP','TFun',...
[-1 -1],[2 1],2,0.0001)
```

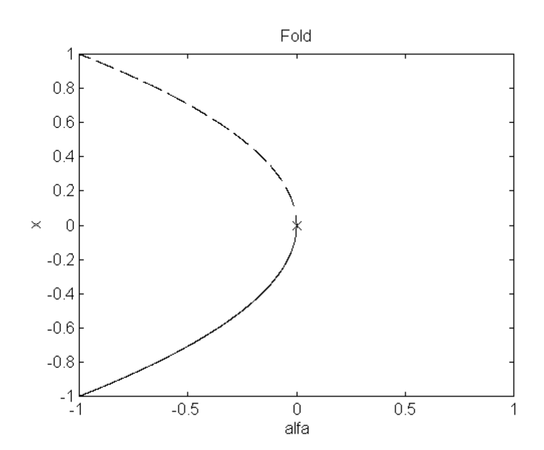


Figura 17 – Función CTCP en el IGB

Los equilibrios estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el equilibrio es indeterminado se grafica en gris (cruz). En el archivo `TFUN.M` se encuentra definida la función de prueba que permite identificar la bifurcación directamente en medio de dos equilibrios. Este resultado se consigna en el archivo `RESULTADOS.TXT`.

```
type resultados.txt

IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
8-May-0

Método: Predictor Tangente Continuo con Parametrización
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo del Jacobiano parametrizado: FoldJP
Archivo de la función de prueba: TFun

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación FOLD entre los equilibrios
Y=[-0.001032; -1.065e-006] e-valor=-0.002064
Y=[0.0003821; -1.46e-007] e-valor=0.0007643

Bifurcación FOLD entre los equilibrios
Y=[0.0003821; -1.46e-007] e-valor=0.0007643
Y=[-0.001032; -1.065e-006] e-valor=-0.002064

Bifurcación FOLD entre los equilibrios
```

```
Y=[-0.001032; -1.065e-006] e-valor=-0.002064
Y=[0.0003821; -1.46e-007] e-valor=0.0007643

Bifurcación FOLD entre los equilibrios
Y=[0.0003821; -1.46e-007] e-valor=0.0007643
Y=[-0.001032; -1.065e-006] e-valor=-0.002064
```

5.7.5 Algoritmo

El equilibrio inicial se calcula con el método de Newton – Raphson multiparamétrico del siguiente modo,

```
MPPAR=length(y0)+1:2*length(y0)+1;
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción secante es,

```
e=y0*0;
e(length(y0))=1;
JZe=[J,Z;e'];
s=inv(JZe)*e;
s=s/norm(s);
y2=y1+h*s;
```

Finalmente la corrección se hace utilizando el método de Newton – Raphson multimparamétrico, con base en la función parametrizada y su Jacobiano, así

```
y2=newtonmp(FUNP,GRADFUNP,[y2',y1',h],MPPAR',DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.7.6 Ver también

CTC, CTCD, CTCP, CTCPD

5.7.7 Referencias

Capítulo 6 .

5.8 CTCPD

5.8.1 Propósito

Localizar bifurcaciones por continuación con predictor tangente con parametrización y corrector basado en el método de Newton – Raphson para sistemas dinámicos discretos.

5.8.2 Sintaxis

```
flag_stop=ctcpd(PLIM,FUN,GRADFUN,FUNP,GRADFUNP,...
 GRADFUND,TFUN,y0,PPLT,PPAR,s,DELTA,EPSILON,PREC)
```

5.8.3 Descripción

`CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y)` realiza el proceso de continuación tangente con parametrización por longitud de arco de la función descrita en el archivo 'FUN' a partir de un punto inicial `Y=[y1,y2,...]` calculando su derivada descrita en la función `'GRADFUN'`.

`PLIM` especifica los límites de la gráfica donde se van a mostrar los puntos fijos y los equilibrios. `PLIM` es una matriz 2x2 o 3x2 que contiene los límites de cada uno de los ejes, en forma de vectores fila. En el caso de una gráfica 2D, contendrá los límites de los ejes horizontal y vertical. Para 3D contendrá además los límites del eje transversal. `PLIM=[h_inf,h_sup; v_inf,v_sup; t_inf,t_sup]`.

`'FUN'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo. Ver el archivo `'FUN.M'`.

`'GRADFUN'` es el nombre del archivo con extensión `.M` donde se encuentra definido el Jacobiano de la función del sistema continuo. Ver el archivo `'GRADFUN.M'`.

`'FUNP'` es el nombre del archivo con extensión `.M` donde se encuentra definida la función del sistema continuo parametrizada. Ver el archivo `'FUNP.M'`.

'GRADFUNP' es el nombre del archivo con extensión .M donde se encuentra definido el Jacobiano de la función del sistema continuo parametrizado. Ver el archivo 'GRADFUNP.M'.

'GRADFUND' es el nombre del archivo con extensión .M donde se encuentra definido el Jacobiano de la función del sistema discreto. Ver el archivo 'GRADFUND.M'.

'TFUN' es el nombre del archivo con extensión .M donde se encuentra definida la función de prueba. Ver el archivo 'TFUND.M'.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT) permite identificar las posiciones de los parámetros que se desea plotear. PPLT es un vector que contiene las posiciones de estas. Si PPLT no es dado, se plotearán las primeras variables.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR) permite identificar las posiciones de los parámetros independientes [a1,a2,...] PPAR es un vector columna que contiene sus posiciones. Si PPAR no está definido, su valor se asume 0 y se considerará que no hay parámetros independientes.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR,S) permite especificar el tamaño y la dirección del paso inicial. S es el tamaño de paso (longitud de arco). Su signo indica hacia donde se da el primer paso. Si S no es dado, se asume como de una centésima del eje horizontal.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR,S,DELTA,EPSILON) permite especificar la tolerancia para localizar las soluciones de 0=FUN(y0), utilizando el criterio de convergencia por tamaño de paso y valor de la función respectivamente. Por defecto estos valores son 10^-5.

CTCP(PLIM,'FUN','GRADFUN','FUNP','GRADFUNP','GRADFUND','TFUN',Y, PPLT,PPAR,S,DELTA,EPSILON,PREC) permite especificar la precisión con la que se desea presentar cada uno de los puntos fijos encontrados.

Con este método no se puede sobrepasar la bifurcación de *fold*, y en general ningún ramal que tenga una recta tangente con pendiente infinita. En este punto la función entra en un *loop*.

5.8.4 Ejemplo

La función,

$$x \mapsto \alpha + x + x^2$$

que describe la forma normal de la bifurcación de *fold* en un sistema dinámico discreto se encuentra descrita en el archivo FOLDD.M, así

```
function yp=foldd(y)
x=y(1); alfa=y(2);
yp=alfa+x+x^2;
```

La función equivalente como sistema dinámico continuo, se encuentra descrita en el archivo FOLD.M, así

```
function yp=fold(y)
x=y(1);alfa=y(2);
yp=alfa+x^2;
```

El Jacobiano de la función discreta se encuentra definido en el archivo FOLDJD.M, así

```
function [J,ValPr]=foldjd(y)
x=y(1); alfa=y(2);
J=1+2*x; ValPr=eig(J);
```

El Jacobiano de la función continua se encuentra definido en el archivo FOLDJ.M, así

```
function [J,ValPr,Z]=foldj(y)
x=y(1);alfa=y(2);
J=2*x;Z=1;ValPr=eig(J);
```

La función continua parametrizada respecto a la longitud de arco se describe en el archivo `FOLDP.M`, así

```
function ypp=foldp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
ypp=[alfa + x^2;(x-x0)^2 + (alfa-alfa0)^2 - h^2];
```

El Jacobiano de esta función continua parametrizada, se describe en el archivo `FOLDJP.M`, así

```
function Jp=foldjp(y)
x=y(1); alfa=y(2); x0=y(3); alfa0=y(4); h=y(5);
J=foldj(y);Z=1;
dP=[2*(x-x0),2*(alfa-alfa0)];
Jp=[J,Z;dP];
```

La bifurcación de *fold* se puede localizar en este sistema mediante la función `CSCPD`, así:

```
ctcpd([-1 1;-1 1],'Fold','FoldJ','FoldP','FoldJP','FoldJD',...
'TFun',[-1 -1],[2 1],2,0.001)
ctcpd([-1 1;-1 1],'Fold','FoldJ','FoldP','FoldJP','FoldJD',...
'TFun',[1 -1],[2 1],2,0.001)
```

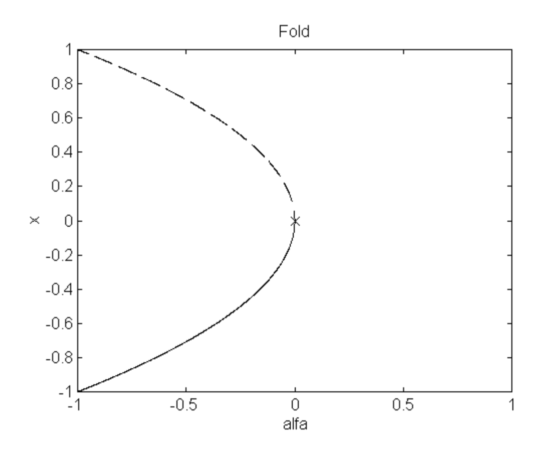


Figura 18 – Función CTCPD en el IGB

Los puntos fijos estables se grafican en azul (línea continua) y los inestables en rojo (línea discontinua). Cuando el punto fijo es indeterminado se grafica en gris (cruz). En el archivo `TFUND.M` se encuentra definida la función de prueba que

permite identificar la bifurcación directamente en medio de dos equilibrios. Este resultado se consigna en el archivo `RESULTADOS.TXT`.

```
type resultados.txt

IDENTIFICADOR GRAFICO DE BIFURCACIONES
ver 5.0

ARCHIVO DE RESULTADOS
9-May-0

Método: Continuación Tangente con Parametrización
Archivo de la función: Fold
Archivo del Jacobiano: FoldJ
Archivo de la función parametrizada: FoldP
Archivo del Jacobiano parametrizado: FoldJP
Archivo de la función de prueba: TFun

Punto inicial Y0=[-1; -1]
Posiciones de los parámetros=2

BIFURCACIONES ENCONTRADAS

Bifurcación FOLD entre los equilibrios
Y=[-0.5; -0.3] e-valor=-0.0009
Y=[-0.5; -0.2] e-valor=0.001
```

5.8.5 Algoritmo

El equilibrio inicial se calcula con el método de Newton – Raphson multiparamétrico del siguiente modo,

```
MPPAR=length(y0)+1:2*length(y0)+1;
y1=newtonmp(FUN,GRADFUN,y0,PPAR,DELTA,EPSILON);
```

El algoritmo utilizado para la predicción secante es,

```
e=y0*0;
e(length(y0))=1;
JZe=[J,Z;e'];
s=inv(JZe)*e;
s=s/norm(s);
y2=y1+h*s;
```

Finalmente la corrección se hace utilizando el método de Newton – Raphson multimparamétrico, con base en la función parametrizada y su Jacobiano, así

```
y2=newtonmp(FUNP,GRADFUNP,[y2',y1',h],MPPAR',DELTA,EPSILON);
```

El algoritmo tiene implementado un control de paso que le permite disminuir su tamaño cerca de los puntos de bifurcación y aumentarlo cuando en las zonas que se encuentran lejos de esta.

5.8.6 Ver también

`CTC, CTCD, CTCP`

5.8.7 Referencias

Capítulo 6 .

6. Funciones de prueba

Los métodos directos para la identificación del tipo de bifurcación se basan en las funciones de prueba desarrolladas para el IGB. Las funciones de prueba pueden ser programadas por el usuario. Para que puedan ser correctamente interpretadas por el IGB, deben retornar como primer valor un *flag* que cumpla:

Valor del *flag* de retorno	**Significado**
0	no hay bifurcación
1	Bifurcación FOLD continua
2	Bifurcación HOPF
3	Bifurcación FOLD discreta
4	Bifurcación FLIP
5	Bifurcación NEIMARK-SACKER
6	bifurcación DESCONOCIDA

Los parámetros de entrada tampoco pueden ser modificados. Para esta versión del IGB, estos parámetros deben ser los valores propios de los equilibrios o puntos fijos en medio de los cuales se desea saber la existencia de una bifurcación.

El usuario es libre de modificar el tratamiento de los parámetros de entrada, pero no puede agregar más. Así mismo el usuario debe conservar nomenclatura de los datos de salida para que el IGB pueda interpretar la bifurcación en el momento de generar el archivo de resultados.

El IGB tiene implementadas dos funciones de prueba por defecto:

- `TFUN`, para sistemas dinámicos continuos
- `TFUND`, para sistemas dinámicos discretos

6.1 TFun

6.1.1 Propósito

Identificar bifurcaciones en sistemas dinámicos continuos de forma directa.

6.1.2 Sintaxis

```
flag_out=tfun(eig1,eig2)
```

6.1.3 Descripción

`TFUN` es la función de prueba para la identificación de bifurcaciones en sistemas dinámicos continuos. Retorna:

0	si no hay bifurcación
1	en bifurcación *fold*
2	en bifurcación de Hopf
6	en bifurcación desconocida

Las funciones de prueba no son de gran utilidad cuando se utilizan por sí solas. En el IGB estas funciones acompañan los métodos de continuación para identificar bifurcaciones en un ramal. Por esta razón las funciones de prueba son poco flexibles en el sentido que sus parámetros de entrada deben ser siempre los valores propios de los equilibrios o puntos fijos `(eig1, eig2)`, y que sus valores de retorno deben ser los establecidos.

6.1.4 Ejemplo

Una bifurcación de *fold* puede identificarse directamente con la función de prueba `TFUN`, así

```
flag_out=tfun(1,-1)
flag_out=1
```

Una bifurcación de Hopf puede identificarse directamente con la función de prueba `TFUN`, así

```
flag_out=tfun([-1+i -1+i],[1+i -1+i])
flag_out=2
```

6.1.5 Algoritmo

```
if isreal(eig1) & isreal(eig2) %si los e-valores son reales
      if (eig1*eig2)<=0
            tf=1; %Bifurcación FOLD
      end
else
      if (sum(real(eig1))*sum(real(eig2)))<=0
            tf=2; %Bifurcación HOPF
      end
end
```

6.1.6 Ver también

TFUND

6.1.7 Referencias

Capítulo 6 .

6.2 TFunD

6.2.1 Propósito

Identificar bifurcaciones en sistemas dinámicos continuos de forma directa.

6.2.2 Sintaxis

```
function tf=tfund(eig1,eig2)
```

6.2.3 Descripción

`TFUND` es la función de prueba para la identificación de bifurcaciones en sistemas dinámicos discretos. Retorna:

0	si no hay bifurcación
3	en bifurcación *fold*
4	en bifurcación *flip*
5	en bifurcación Neimark - Sacker
6	en bifurcación desconocida

Las funciones de prueba no son de gran utilidad cuando se utilizan por sí solas. En el IGB estas funciones acompañan los métodos de continuación para identificar bifurcaciones en un ramal. Por esta razón las funciones de prueba son poco flexibles en el sentido que sus parámetros de entrada deben ser siempre los valores propios de los equilibrios o puntos fijos `(eig1, eig2)`, y que sus valores de retorno deben ser los establecidos.

6.2.4 Ejemplo

Una bifurcación de *fold* en un sistema dinámico discreto puede identificarse directamente con la función de prueba `TFUND`, así

```
flag_out=tfund(1.1,0.9)
flag_out=3
```

Una función de *flip* así,

```
flag_out=tfund(-1.1,-0.9)
flag_out=4
```

Finalmente una función de Neimark – Sacker puede identificarse directamente así por ejemplo,

```
eig1=[0.9*cos(pi/4)+i*0.9*sin(pi/4) -0.9*cos(pi/4)+i*0.9*sin(pi/4)]
eig2=[1.1*cos(pi/4)+i*1.1*sin(pi/4) -1.1*cos(pi/4)+i*1.1*sin(pi/4)]
flag_out=tfund(eig1,eig2)
flag_out=5
```

6.2.5 Algoritmo

```
if ~prod(size(eig1)==size(eig2))
      tf=6; %bifurcación DESCONOCIDA
else
      if (norm(eig1(1))<1 & norm(eig2(1))>=1) | (norm(eig1(1))>=1 &
norm(eig2(1))<1)
            if isreal(eig1) & isreal(eig2)
                  if eig1>0 & eig2>0
                        tf=3; %Bifurcación FOLD dis
                  elseif eig1<0 & eig2<0
                        tf=4; %Bifurcación FLIP
                  else
                        tf=6; %Bifurcación DESCONOCIDA
                  end
            elseif ~isreal(eig1) & ~isreal(eig2)
                  tf=5; %Bifurcación de NEIMARK-SACKER
            else
                  tf=6; %Bifurcación desconocida
            end
      end
end
```

6.2.6 Ver también

TFUN

6.2.7 Referencias

Capítulo 6 .

7. Funciones del sistema dinámico

El sistema dinámico a analizar, debe ser descrito de una forma tal que pueda ser interpretado por IGB. Las funciones del sistema dinámico describen son la base para la identificación de las bifurcaciones. Estas funciones deben ser creadas por el usuario con base en las plantillas creadas para este propósito.

Las plantillas de las funciones de los sistemas dinámicos continuos son:

- Fun
- GradFun
- FunP
- GradFunP

Para el análisis de sistemas dinámicos discretos se utilizan, además de las plantillas de los sistemas continuos, las siguientes,

- FunD
- GradFunD

7.1 Fun

7.1.1 Propósito

Con base en esta plantilla se crea la función que describe el sistema dinámico continuo.

7.1.2 Sintaxis

```
YP=FUN(Y)
```

7.1.3 Descripción

En este archivo se describe el sistema continuo con parámetros independientes dado por la ecuación,

$$\dot{x} = f(x, \alpha), x \in \mathbf{R}^n, \alpha \in \mathbf{R}^m$$

El vector de entrada (x, α) es de dimensión $n+m$, y contiene n variables de estado y m parámetros independientes.

El vector de respuesta $\dot{x}$ es de dimensión n, y contiene el estado de las n variables de estado.

7.1.4 Ejemplo

La manera en la que se presenta el sistema dinámico continuo que describe la forma normal de la bifurcación de Neimark – Sacker,

$$f(x, \alpha) = \begin{pmatrix} \cos\theta(\alpha) & -\text{sen}\,\theta(\alpha) \\ \text{sen}\,\theta(\alpha) & \cos\theta(\alpha) \end{pmatrix} \left[(1+\alpha)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left(x_1^2 + x_2^2\right)\begin{pmatrix} a(\alpha) & -b(\alpha) \\ b(\alpha) & a(\alpha) \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right] - \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

se describe en el archivo `SACKER.M` de así,

```
function yp=sacker(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=-alfa; b=0;
A=[cos(teta) -sin(teta);sin(teta) cos(teta)];
B=(1+alfa)*[x1;x2];
C=(x1^2+x2^2)*[a -b;b a]*[x1;x2];
yp=A*(B+C)-[x1;x2];
```

7.1.5 Ver también

`GRADFUN, FUNP, GRADFUNP, FUND, GRADFUND`

7.1.6 Referencias

Capítulos 3 y 4

7.2 GradFun

7.2.1 Propósito

Con base en esta plantilla se crea la el Jacobiano de la función que describe el sistema dinámico continuo.

7.2.2 Sintaxis

```
[J,Z,ValPr]=GRADFUN(Y)
```

7.2.3 Descripción

En este archivo se describe el Jacobiano de la función continua del sistema dinámico en estudio.

$$J = \frac{d}{dx} f(x, \alpha),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m$$

El Jacobiano J es una matriz n x n que evalúa las derivadas de las funciones del vector de funciones que describen el sistema dinámico continuo respecto a las n variables de estado x.

La matriz Z de dimensión n x m consigna las derivadas de la función continua respecto a los parámetros independientes,

$$Z = \frac{d}{d\alpha} f(x, \alpha),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m$$

`ValPr` consigna los valores propios de J evaluados en (x, α).

7.2.4 Ejemplo

El Jacobiano de la función que describe la forma normal de la función de Neimark – Sacker, es la derivada de la función $f(x, \alpha)$ respecto a x,

$$f(x, \alpha) = \begin{pmatrix} \cos\theta(\alpha) & -\operatorname{sen}\theta(\alpha) \\ \operatorname{sen}\theta(\alpha) & \cos\theta(\alpha) \end{pmatrix}$$

$$\left[(1+\alpha)\begin{pmatrix}x_1\\x_2\end{pmatrix}+\left(x_1^2+x_2^2\right)\begin{pmatrix}a(\alpha) & -b(\alpha)\\ b(\alpha) & a(\alpha)\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}\right]-\begin{pmatrix}x_1\\x_2\end{pmatrix}$$

$$J=\frac{d}{dx}f(x,\alpha)$$

Y se presenta del siguiente modo en el archivo `SACKERJ.M`,

```
function [J,ValPr,Z]=sackerj(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=-alfa; b=0;
dAdx1=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(a*x1-b*x2);
dAdx2=   0    +   (x1^2+x2^2)*(-b)+ (2*x2)*(a*x1-b*x2);
dBdx1=   0    +   (x1^2+x2^2)*(b)+  (2*x1)*(b*x1+a*x2);
dBdx2=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(b*x1+a*x2);
I=[1 0;0 1];
J=[cos(teta)*dAdx1-sin(teta)*dAdx1, cos(teta)*dAdx2-sin(teta)*dAdx2;
   sin(teta)*dBdx1+cos(teta)*dBdx1, cos(teta)*dBdx2-sin(teta)*dBdx2]
   -I;
dAdalfa=x1;dBdalfa=x2;
Z=[cos(teta)*dAdalfa-sin(teta)*dAdalfa,...
   sin(teta)*dBdalfa+cos(teta)*dBdalfa];
ValPr=eig(J);
```

7.2.5 Ver también

FUN, FUNP, GRADFUNP, FUND, GRADFUND

7.2.6 Referencias

Capítulos 3 y 4

7.3 FunP

7.3.1 Propósito

Con base en esta plantilla se crea la función que describe el sistema dinámico continuo parametrizado respecto a la longitud de arco.

7.3.2 Sintaxis

```
YPP=FUNP(Y)
```

7.3.3 Descripción

En este archivo se describe el sistema continuo con parámetros independientes parametrizado respecto a la longitud de arco,

$$\dot{x}p = fp\ (x,\ \alpha,\ x_0,\ \alpha_0,\ s);\ x,\ x_0 \in \mathbf{R}^n,\ \alpha\ ,\alpha_0 \in \mathbf{R}^m,\ s \in \mathbf{R}$$

El vector de entrada está formado por n variables de estado x, m parámetros independientes α, n condiciones iniciales de las variables es estado x, m condiciones iniciales de los parámetros independientes, y un parámetros adicional s que es la longitud de arco.

El vector de respuesta $\dot{x}p$ es de dimensión n+m, y contiene el estado de las n variables de estado x y las m funciones adicionales que parametrizan la función respecto a la longitud de arco s. La función de parametrización es,

$$p(x,\ \alpha,\ s) = \sum_{i=1}^{n} (x^i - x_0^i)^2 + \sum_{j=1}^{m} (\alpha^j - \alpha_0^j)^2 - s$$

La función parametrizada queda definida así,

$$\dot{x}p = \begin{bmatrix} \dot{x} \\ p \end{bmatrix}$$

donde $\dot{x} = f\,(x,\ \alpha)$.

7.3.4 Ejemplo

La función que describe la forma normal de la bifurcación de Neimark – Sacker se parametriza respecto a la longitud de arco agregando el polinomio

$$p(x, \alpha, s) = (x - x_0)^2 + (\alpha - \alpha_0)^2 - s$$

a la función $f(x, \alpha)$

$$f(x, \alpha) = \begin{pmatrix} \cos\theta(\alpha) & -\operatorname{sen}\theta(\alpha) \\ \operatorname{sen}\theta(\alpha) & \cos\theta(\alpha) \end{pmatrix} \left[(1+\alpha)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left(x_1^2 + x_2^2\right)\begin{pmatrix} a(\alpha) & -b(\alpha) \\ b(\alpha) & a(\alpha) \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right] - \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

así,

$$fp(x, \alpha) = \begin{bmatrix} f \\ p \end{bmatrix}$$

Esta función parametrizada se presenta en el archivo `SACKERP.M` de la siguiente forma,

```
function ypp=sackerp(y)
x1=y(1); x2=y(2); alfa=y(3);
x10=y(4); x20=y(5); alfa0=y(6); s=y(7);
teta=pi/2; a=-alfa; b=0;
yp=sacker(y);
p=(x1-x10)^2+(x2-x20)^2+(alfa-alfa0)^2-s;
ypp=[yp;p];
```

7.3.5 Ver también

`FUN, GRADFUN, GRADFUNP, FUND, GRADFUND`

7.3.6 Referencias

Capítulo 6

7.4 GradFunP

7.4.1 Propósito

Con base en esta plantilla se crea el Jacobiano de la función que describe el sistema dinámico continuo parametrizada respecto a la longitud de arco.

7.4.2 Sintaxis

```
JP=GRADFUNP(Y)
```

7.4.3 Descripción

En este archivo se describe el Jacobiano de la función continua del sistema dinámico en estudio parametrizada.

$$Jp = \frac{d}{dx} fp(x, \alpha, x_0, \alpha_0, s);\ x, x_0 \in \mathbf{R}^n,\ \alpha, \alpha_0 \in \mathbf{R}^m,\ s \in \mathbf{R}$$

Es decir,

$$Jp = \begin{bmatrix} J\ Z \\ \hline dp \end{bmatrix}$$

El Jacobiano de la función parametrizada Jp es una matriz $(n+m)$ x $(n+m)$ que evalúa las derivadas de las funciones del vector de funciones que describen el sistema dinámico continuo parametrizado.

El Jacobiano de la función continua J es una matriz n x n que las derivadas de las funciones del vector de funciones que describen el sistema respecto a las n variables de estado.

$$J = \frac{d}{dx} f(x, \alpha),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m$$

La matriz Z de dimensión n x m consigna las derivadas de la función continua respecto a los parámetros independientes,

$$Z = \frac{d}{d\alpha} f(x,\ \alpha),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m$$

La matriz dp de dimensión m x $(n+m)$ consigna las derivadas de la función de parametrización $p(x,\ \alpha,\ s)$, respecto a las variables de estado y los parámetros independientes

$$dp = \frac{d}{d(x,\alpha)} p(x,\ \alpha,\ s),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m,\ s \in \mathbf{R}$$

7.4.4 Ejemplo

Para la forma norma normal de la bifurcación de Neimark – Sacker, se describe el Jacobiano de la función parametrizada así,

$$Jp = \left[\frac{J\ Z}{dp}\right]$$

donde las derivadas de la función J y Z respecto a las variables de estado y a los parámetros independientes se presentan como,

$$J = \frac{d}{dx} f(x,\ \alpha),\quad Z = \frac{d}{d\alpha} f(x,\ \alpha)$$

$$f(x,\ \alpha) = \begin{pmatrix} \cos\theta(\alpha) & -\operatorname{sen}\theta(\alpha) \\ \operatorname{sen}\theta(\alpha) & \cos\theta(\alpha) \end{pmatrix}$$

$$\left[(1+\alpha)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left(x_1^2 + x_2^2\right)\begin{pmatrix} a(\alpha) & -b(\alpha) \\ b(\alpha) & a(\alpha) \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right] - \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Asimismo se presenta la derivada de la función de parametrización dp como,

$$dp = \frac{d}{d(x,\alpha)} p(x,\ \alpha,\ s)$$

$$p(x, \alpha, s) = (x - x_0)^2 + (\alpha - \alpha_0)^2 - s$$

El Jacobiano parametrizado se consigna en el archivo `SACKERJP.M` así,

```
function Jp=sackerjp(y)
x1=y(1); x2=y(2); alfa=y(3);
x10=y(4); x20=y(5); alfa0=y(6); s=y(7);
teta=pi/2; a=-alfa; b=0;
[J,ValPr,Z]=sackerj(y);
dP=[2*(x1-x10) 2*(x2-x20) 2*(alfa-alfa0)];
Jp=[J,Z;dP];
```

7.4.5 Ver también

`FUN, GRADFUN, FUNP, FUND, GRADFUND`

7.4.6 Referencias

Capítulo 6

7.5 FunD

7.5.1 Propósito

Con base en esta plantilla se crea la función que describe el sistema dinámico discreto.

7.5.2 Sintaxis

```
YP=FUND(Y)
```

7.5.3 Descripción

En este archivo se describe el sistema continuo con parámetros independientes dado por la ecuación,

$$x \mapsto f(x, \alpha),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m$$

El vector de entrada (x, α) es de dimensión n+m, y contiene n variables de estado y m parámetros independientes.

El vector de respuesta x es de dimensión n, y contiene el estado de las n variables de estado.

7.5.4 Ejemplo

La manera en la que se presenta el sistema dinámico discreto que describe la forma normal de la bifurcación de Neimark – Sacker,

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} \cos\theta(\alpha) & -\operatorname{sen}\theta(\alpha) \\ \operatorname{sen}\theta(\alpha) & \cos\theta(\alpha) \end{pmatrix} \left[(1+\alpha)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left(x_1^2 + x_2^2\right)\begin{pmatrix} a(\alpha) & -b(\alpha) \\ b(\alpha) & a(\alpha) \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right]$$

se describe en el archivo `SACKER.M` de así,

```
function yp=sackerd(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=-alfa; b=0;
A=[cos(teta) -sin(teta);sin(teta) cos(teta)];
B=(1+alfa)*[x1;x2];
```

```
C=(x1^2+x2^2)*[a -b;b a]*[x1;x2];
yp=A*(B+C);
```

7.5.5 Ver también

FUN, GRADFUN, FUNP, GRADFUNP, GRADFUND

7.5.6 Referencias

Capítulo 4

7.6 GradFunD

7.6.1 Propósito

Con base en esta plantilla se crea la el Jacobiano de la función que describe el sistema dinámico discreto.

7.6.2 Sintaxis

```
[J,ValPr]=GRADFUND(Y)
```

7.6.3 Descripción

En este archivo se describe el Jacobiano de la función continua del sistema dinámico en estudio.

$$J = \frac{d}{dx} f(x, \alpha),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m$$

El Jacobiano J es una matriz n x n que evalúa las derivadas de las funciones del vector de funciones que describen el sistema dinámico continuo respecto a las n variables de estado x.

La matriz Z de dimensión n x m consigna las derivadas de la función continua respecto a los parámetros independientes,

$$Z = \frac{d}{d\alpha} f(x, \alpha),\ x \in \mathbf{R}^n,\ \alpha \in \mathbf{R}^m$$

`ValPr` consigna los valores propios de J evaluados en (x, α).

7.6.4 Ejemplo

El Jacobiano de la función que describe la forma normal de la función de Neimark – Sacker, es la derivada de la función $f(x, \alpha)$ respecto a x,

$$f(x,\,\alpha)\;=\begin{pmatrix}\cos\,\theta(\alpha) & -\,\mathrm{sen}\;\theta(\alpha)\\ \mathrm{sen}\;\theta(\alpha) & \cos\,\theta(\alpha)\end{pmatrix}\left[(1+\alpha)\begin{pmatrix}x_1\\ x_2\end{pmatrix}+\left(x_1^2+x_2^2\right)\begin{pmatrix}a(\alpha) & -b(\alpha)\\ b(\alpha) & a(\alpha)\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}\right]$$

$$J=\frac{d}{dx}f(x,\,\alpha)$$

Y se presenta del siguiente modo en el archivo `SACKERJD.M`,

```
function [J,ValPr]=SackerJD(y)
x1=y(1); x2=y(2); alfa=y(3);
teta=pi/2; a=-alfa; b=0;
dAdx1=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(a*x1-b*x2);
dAdx2=   0    +   (x1^2+x2^2)*(-b)+ (2*x2)*(a*x1-b*x2);
dBdx1=   0    +   (x1^2+x2^2)*(b)+  (2*x1)*(b*x1+a*x2);
dBdx2=(1+alfa)+   (x1^2+x2^2)*(a)+  (2*x1)*(b*x1+a*x2);
J=[cos(teta)*dAdx1-sin(teta)*dAdx1, cos(teta)*dAdx2-sin(teta)*dAdx2;
  sin(teta)*dBdx1+cos(teta)*dBdx1,cos(teta)*dBdx2-sin(teta)*dBdx2];
ValPr=eig(J);
```

7.6.5 Ver también

`FUN, GRADFUN, FUNP, GRADFUNP, FUND`

7.6.6 Referencias

Capítulo 4